Disappearing Mobile Devices

Tao Ni[1, 2] and Patrick Baudisch[1]

[1]Hasso Plattner Institute
Potsdam, Germany

[2]Department of Computer Science
Virginia Tech, Blacksburg, VA, USA

nitao@cs.vt.edu, patrick.baudisch@hpi.uni-potsdam.de

ABSTRACT

In this paper, we extrapolate the evolution of mobile devices in one specific direction, namely miniaturization. While we maintain the concept of a device that people are aware of and interact with intentionally, we envision that this concept can become small enough to allow invisible integration into arbitrary surfaces or human skin, and thus truly ubiquitous use. This outcome assumed, we investigate what technology would be most likely to provide the basis for these devices, what abilities such devices can be expected to have, and whether or not devices that size can still allow for meaningful interaction. We survey candidate technologies, drill down on gesture-based interaction, and demonstrate how it can be adapted to the desired form factors. While the resulting devices offer only the bare minimum in feedback and only the most basic interactions, we demonstrate that simple applications remain possible. We complete our exploration with two studies in which we investigate the affordance of these devices more concretely, namely marking and text entry using a gesture alphabet.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces: Input Devices and Strategies, Interaction Styles.

General terms: Design**,** Human Factors.

Keywords: Miniaturization, mobile device, input device, sensor, wearable, ubicomp, interaction technique, gesture.

INTRODUCTION

In this paper, we investigate the limits of the miniaturization of mobile devices, researching what the smallest future devices might be, and how users would interact with them.

In the past, the miniaturization of mobile devices has progressed from notebook computers and PDAs to increasingly smaller devices, such as interactive watches [1] and rings [44, 8]. At first sight, it might therefore appear that a continuing miniaturization process is destined to let devices shrink past any specific size limit. This is not the case, however. What constraints miniaturization is the devices' *user interface* hardware, the size of which is linked to *human* constraints. Screens, for example, have to be large enough to be seen, keyboards large enough to be typed on.

Since human constraints are largely constant (e.g.,*fat finger problem* [1, 42]), arbitrary hardware miniaturization does not directly lead to arbitrarily small devices, but to devices the sizes and shapes of which are determined by their user interface hardware. For larger devices that has already happened: the size of current notebook computers is already largely determined by screen diagonal and keyboard size. As non-user interface hardware continues to shrink, this observation will apply to increasingly smaller devices.

UIST'09, October 4–7, 2009, Victoria, British Columbia, Canada.

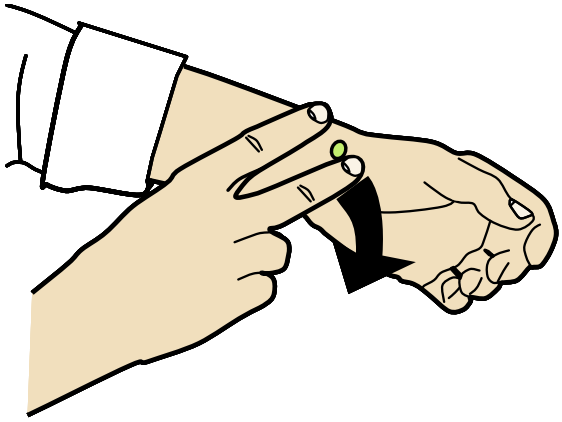

Figure 1: *Disappearing mobile devices* are too small to be held, so they have to be mounted, e.g. on the user's wrist. Here the user is entering a '2' by *scanning* two fingers. Depending on the desired degree of miniaturization, *scanning* causes it to perceive a sequence of gestures, marks, or just Morse code.

Exploring the miniaturization of future mobile devices therefore means to explore the miniaturization of user interface hardware, more specifically to examine how to strip devices of any user interface hardware that requires physical extent.

As a first step, this requires removing all interface hardware that is linked to the user's finger size, eyesight, or other human constraints. Instead, alternative approaches have to be used, such as gesture-based interaction, as previously explored by projects such as *FlowMouse* [46] or *Gesture Pendant* [41]. As a second step, we also need to remove all interface hardware the physical properties of which resist miniaturization. The miniaturization of gesture pendant, for example, is limited by the size of the optical path inside the camera and the necessity to illuminate a large space (here 36 infrared LEDs).

In this paper, we explore gesture interaction *on the surface* of the device as one path to ultimate miniaturization. Compared to other gesture-based approaches surface-based interaction requires only minimal illumination and can be accomplished with particularly small sensors. Based on this design, we reconsider the basics of gesture interaction.

The resulting "ultimately miniaturized" mobile devices adopt properties from wearable and ubiquitous computing: They can be worn and are always in reach; they offer only simple interactive capabilities and blend invisibly into any surface, almost like the sensors/actuators in smart rooms [2]. However, the "infrastructure" the device blends into is the user's clothing or skin rather than a room (Figure 2). Nonetheless, we envision these devices to still be "traditional" mobile devices in that they are operated intentionally (unlike ubicomp [43]) and afford manual interaction.

Based on these properties and inspired by *disappearing computers* [36] in ubicomp, we refer to these future devices as *disappearing mobile devices.*

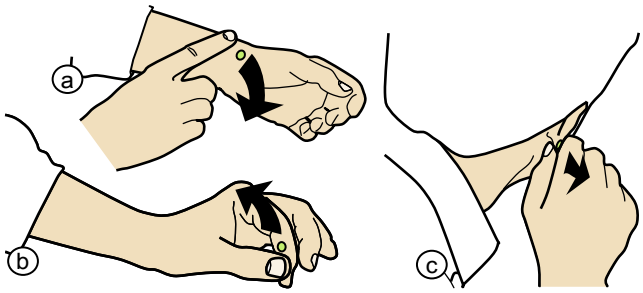

Figure 2: Examples of disappearing mobile devices that are worn or implanted (a) in the wrist, providing visual feedback, (b) on the finger, providing tactile feedback, and (c) in the earlobe, providing auditory feedback.

Today's technology does not allow us to create such devices "to scale". Neither can we anticipate in which particular order hardware components will shrink or what particular sequence of devices will result from the evolution of hardware. We therefore instead focus on analyzing candidate technologies with respect to their *theoretical* limits. We then explore interaction techniques for such devices. While we recognize that output is equally important and challenging, we focus our exploration almost exclusively on input. We complete our exploration with two studies in which we investigate the affordance of these devices more concretely, namely interaction based on marking and text entry using a gesture alphabet.

RELATED WORK

This work builds on mobile and wearable computing as well as interaction with very small devices.

Mobile and Wearable Computing

While *mobile computing* initially focused on hand-held devices, such as PDAs and phones (e.g., [1, 42, 45]), continuous miniaturization has allowed researchers to start exploring devices small enough that they can be worn. The IBM Linux Watch [31] runs Linux. IBM's digital jewelry consists of earrings, a pendant and a Trackpoint ring [28]. *Telebeads* [19] are mobile mnemonic artifacts that allow teenagers to link individuals or groups with wearable objects such as handmade jewelry. Fukumoto's finger-ring shaped bone conduction HANDset [8, 10] allows users to issue fingertip tapping commands. So does *Whisper* [9]. *GestureWrist* [36], worn like a wristband, supports forearm gestures. *GesturePad* hides under clothes [36].

Early work in *wearable computing* focused on augmented reality, in an attempt to reproduce desktop functionalities in a mobile form (e.g., smart clothing [26], augmented reality systems [6]). Later research directed attention to low-level interaction techniques [27], and application scenarios of wearable computing [5, 20].

Interaction with Very Small Devices

Miniaturization has resulted in many challenges for user interface research.

With decreasing device size visual output becomes increasingly limited. Screens have to be small to fit the device, which eventually conflicts with the visual abilities of users [38]. Consequently, some very small devices have abandoned screens entirely in favor of auditory (e.g., *earPod* [52]) or tactile feedback [23]. Mounting the device close to the user's ear (Figure 2c) allows for effective auditory feedback from a small form factor and low power consumption.

Audio is an interesting alternative as well for input, e.g., in the form of speech recognition [40]. The inherent volume of speech input can limit its applicability in situations where others are around [40].

Manual input faces similarly hard challenges. *Text entry:* Keypads with very few keys have been studied in research [25]. Our work in this paper borrows from pen-based techniques, such as *Unistroke* [11], the more mnemonic *Graffiti* [24], and *EdgeWrite* [49], a unistroke-like language that offers extra supports for people with motor impairments. A trackball version uses *target crossing* [47] instead.

Pointing: Tiny rate-controlled joysticks allow for compact form factors [28]. Touchscreens, in contrast, offer position input, which is generally considered a more convenient and efficient way to point. Touch input, however, is complicated by the *fat finger problem* [42], which results from the softness of the user's fingertip and the occlusion of screen contents by the user's finger. The smaller the device the more severe the impact of the fat finger problem [1].

Styli have been used to alleviate this problem [50], as well as techniques that offset the finger with respect to the target (*offset cursor* [34]) or the target with respect to the finger (*shift* [42]). Even with *shift*, front-side touch fails for screen diagonals below 1" [1]. Alternatives are pointing input next to the screen (*SideSight* [3]) or on the device backside (e.g., *LucidTouch* [45], *nanoTouch* [1]).

As mentioned in the introduction, freehand gesture input alleviates some of the size limitations by moving the interaction away from the surface into the space surrounding the device. Liess et al. present a 2D displacement motion sensor based on laser self-mixing [22]. *Gesture pendant* [41] uses a small infrared camera worn as a part of a necklace or pin to recognize various finger and hand gestures. *FlowMouse* [46] uses a webcam to enable pointing, rotation, panning, and marking. Unlike earlier techniques, it increases reliability by using optical flow, rather than hand recognition.

Gesture input in mid air faces three limitations that complicate miniaturization. First, it involves macroscopic cam-

eras. Second, it requires substantial illumination [41]. And third, it can make it hard for users to trigger discreet events, which are important for delimiting gesture languages [14].

HARDWARE FOR DISAPPEARING MOBILE DEVICES

As discussed earlier, camera-based gesture interaction is the starting point for our exploration of arbitrarily small devices, but it has limitations. In this section, we survey existing interface devices, discuss how they might fit into a disappearing device, and what modifications are necessary.

If a technology depends on human factors, such as finger size or eyesight, we scale it to the size of a disappearing mobile device, i.e., size zero. At this point, users cannot resolve the spatial properties of the hardware anymore. All interface elements that are usually spatially distinct, such as pixels, collapse into the same location. A screen of any resolution, for example, thereby becomes indistinguishable from a single pixel. A keyboard becomes functionally equivalent to a single button, as any number of buttons will always be pressed at the same time. We call the result, i.e., hardware with no inherent spatial properties *monolithic*.

In addition, we investigate technological constraints. How far a specific technology will be able to scale depends at least in part on industrial efforts. In some case, however, there are limits resulting from physical principles that can already be predicted today.

Output

Several output modalities make interesting candidates. A key requirement is that the device produces a signal strong enough to be perceivable by human senses. Mounting the device closer to the user's perceptual organs, however, generally reduces the required power and allows for further miniaturization (Figure 2).

When miniaturized, *screens* turn into a single pixel or LED (Figure 2a). Depending on ambient lighting conditions, visual output can be more perceivable than audio or tactile. LEDs can also be manufactured at a very small scale. For example, the 1.0 x 0.8 x 0.2mm PicoLED built by Rohm (www.rohm.com) is claimed to be the world's tiniest diode. The 1.7 x 1.5 x 0.5mm LXCL-PWT1 from Luxeon (www.luxeon.com) is one of the smallest LED flashlights, which emits 26 lumens of light at 350mA.

Audio on a disappearing mobile device can offer only a single channel. Higher sound frequencies and lower volume allow for smaller membranes, which are required for miniaturization. Power requirements can be reduced further by mounting the device close to the user's ears or a bone connected to it (Figure 2c) [21, 8].

Tactile feedback is similar in nature to auditory feedback in that lower frequency and stronger signal require a larger vibrating mass. Humans are generally less sensitive to touch than to sound, but tactile feedback may be viable if mounted directly onto a sensitive skin area (Figure 2b).

In theory, other modalities could be used as well. That said, other skin perceptions, such as heat perception tend to be low in bandwidth. Smell and taste involve chemicals, making them difficult to handle.

Gesture and Touch Input

Since camera-based interaction is limited in terms of power requirements for the illuminant and physical size of the camera, we reduce power requirements by moving towards the surface. We primarily look at interactions that take place in direct physical contact with the device, i.e., variations of touch. At size zero, touch techniques collapse into the following three interaction styles.

Touch: In its simplest form, a device can tell whether it is being touched on not. Implementations include capacitive sensors and thresholded light sensors. Hudson [17] demonstrated how to turn unmodified LED arrays into touch sensitive input devices. Other mechanisms are possible: microphones, for example, respond to touch as long as the hand stays in motion (scratching [13]).

Pressure: Humans are able to operate physical buttons of about 1mm in diagonal. An interaction similar to physical buttons can be implemented using pressure sensors. Ramos et al. show that users can control up to six levels of pressure [35]. Sensing pressure gets harder with decreasing device size, as the force that can be sensed is the product of pressure and surface. Another potential limitation is the fact that sensing pressure requires the device to be mounted on a hard surface in order to allow users to build up pressure.

Motion: Not all devices that sense motion continue to work when scaled to size zero. Capacitive touch pads, for example, turn into touch sensors and cannot track motion anymore. Friction imposes lower size limits on mechanical solutions such as trackballs. A wide range of devices can track *optical* flow. While they still use a camera, limiting the input to the immediate surface of the device reduces the need for illumination substantially. The *Xybernaut Poma* (www.xybernaut.com), for example, is basically an upside-down optical mouse (Figure 7b). Even more power and space-efficient, laser mice use an infrared laser diode instead of an LED. By analyzing the laser speckle pattern they allow for simpler structure, higher accuracy, and capability of working on a wide range of surfaces [33]. Thus, laser technology is a particularly interesting candidate for disappearing mobile devices.

Self-motion sensing using accelerometers (e.g., [16]) is another applicable approach if the device can be mounted such that it can be moved freely (e.g., Figure 2a, but not Figure 2c) and if inadvertent activation can be avoided. Many other touch-based techniques, such as bending [39] and multi-touch [45] do not transfer to miniaturized hardware, because they require more than one contact point, and thus physical extent.

Discussion

Because of the limitations of pressure, we focus on touch and motion. The more powerful choice is clearly motion sensing, but even with the reduced requirement for illumination, optical solutions do have elements that cannot be miniaturized arbitrarily, such as an array of optical sensors and optics that require a certain focal length (a Misumi *MO-R803*, for example, measures 4.4mm in diameter, www.misumi.com.tw).

Overall, devices that sense optical flow clearly have to be considered. However, for the ultimate limits of miniaturization, such as when designing devices to be implanted into human skin, we might have to rely on even smaller input devices, such as touch sensors.

Three Classes of Devices

Based on this discussion, we initially defined two specific models of monolithic input hardware.

A motion scanner consists of a touch sensor combined with a motion sensor. It can perceive *gestures* as well as a distinct *out-of-range* state.

A touch scanner consists of only a single touch sensor (of any underlying technology). It senses two states: *touch* and *out-of-range*.

Due to the limitations of the motion scanner, we introduced a third device of intermediate size and functionality.

A direction scanner consists of three very closely collocated touch sensors mounted in a non-collinear layout. While the touch sensors are too close to be operated individually, directional gestures across the device result in run-time differences between the touch sensors, allowing it to sense the direction of gestures.

The reason we called them scanners is because of the particular interaction style they support. We discuss this interaction style in the following.

INPUT ON MONOLITHIC DEVICES: SCANNING

In order to allow users to interact with these three device types we need to offer a command language, i.e., a set of code words (commands and parameters) and delimiters [14] that allow the device to parse the input stream into individual command and parameter tokens.

For each device type, we begin by discussing delimiters, a crucial element of any gesture language [14]. Then we define other data types: Boolean, integer, float, and character.

The Input Language of Touch Scanners is *Morse*

On touch scanners, the only way to define a delimiter is by means of a timeout. Example for such timeout delimiters can be found in Morse code [29]. Morse uses pauses of different lengths to delimit parts of a character (pause 1 unit), characters (3 units), and words (7 units). The tapping actions on Fukumoto's handset implement Morse [8, 10].

Moving from there, a simple and coherent way to define characters/text on a touch scanner is to adopt Morse code as a whole; it defines characters a sequence of short and long signals. For other applications, we can redefine the meaning of individual codes, but the nature of the interaction remains equivalent to Morse code.

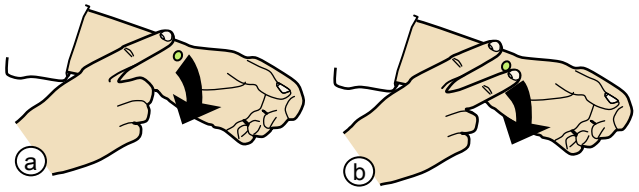

Figure 3: (a) Enter a "1" and (b) a "2" into a touch scanner using a "scanning" interaction.

Figure 3 illustrates an alternative of entering certain parameters that makes use of the specific properties of the device. Rather than tapping codes into the device, users can enter certain codes, such as the numbers from 1 to 4 by *scanning* a hand with the respective number of fingers across the device (Figure 3). Note that this approach may allow scanning more complex objects, such as simplified punch cards.

The Input Language of Direction Scanners is *Marking*

Direction scanners can do everything that touch scanners can. Any language defined for a touch scanner will therefore also run on a direction scanner. In addition, direction scanners support directional gestures, which multiplies the possibilities and enables higher bandwidth than Morse.

The choice of delimiter defines the language. If we define our delimiter to be *lift-off* (transition from *touch* to *out-of-range*), we obtain 8 code words (N, NE, E…) and our language is *marking* [18]. If we use a specific mark or a timeout as a delimiter, code words can be sequences of marks and our resulting language is *simple marks* [51]. Distinguishing marks performed with different numbers of fingers (Figure 3) offers a particularly effective way of entering *simple mark* words based on repetitive strokes.

The Input Language of Motion Scanners is *Unistroke*

Motion scanners can do everything that directional scanners can. In addition, they recognize the path of a gesture.

The language of motion scanners is similar to direction scanners, yet the individual gestures are more powerful. Individual gestures can now, for example, be *compound marks* [51] or character gestures from a *unistroke*-based alphabet, such as *Jot* [11] or *Graffiti* [12] (Figure 4a).

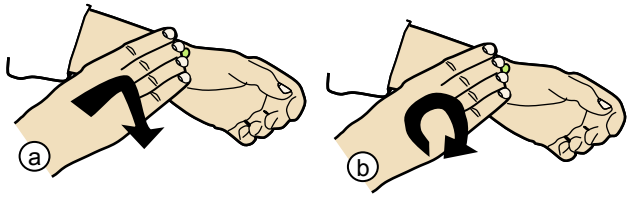

Figure 4: (a) Entering the *Graffiti* character "A" on a motion scanner. (b) Entering a real-value parameter using dialing. Using the hand instead of a finger allows for larger gesture amplitude.

In theory, any gesture that can be completed on a touchscreen with a stylus or finger can also be completed on a motion scanner including continuous gestures, such as rubbing [32] or dialing [30] (Figure 4b). However, there are several unusual properties that arise when applying these techniques to the hand-over-device concept of disappearing mobile devices. We discuss some of these particularities in the following, others in the user study sections.

(Some of) the Particularities of Scanning

Scanning inherits from touch and gestures, and as a result, has several interesting properties.

Unlike gestures in mid-air, scanning allows getting out-of-range very easily. The reason is that scanning takes place near the device surface, which means to get out-of-range, users only need to extend their gesture beyond the field of

view of the sensor. As a result, out-of-range makes a great delimiter that mid-air gesturing does not offer.

On the other hand, the proximity of the user's fingers to the device also limits the amplitude of gestures. If we define out-of-range to function as a delimiter, moving the hand too far (even without lifting it) will accidentally uncover the sensor and thus commit the current gesture prematurely, generally resulting in an error.

Another way of looking at this is to consider scanning as "upside down" mouse input: during scanning, fingers do not act as the pointing device/mouse anymore, but as the mouse pad. In this new role, fingers have to provide enough surface area for the interaction to take place. Small fingers now limits the amplitude of the gestures, the same way that a small mouse pad limits the range of mouse input.

This effect is amplified by the fact that users cannot see the device while scanning, because it is occluded by the user's hand. This increases the risk of accidentally leaving the field of view of the sensor and delimiting prematurely.

In homage to the similarly motivated *fat finger problem*, we termed this problem *small finger problem*. One way of alleviating it is to scan with a larger object, such as an entire hand (Figure 5).

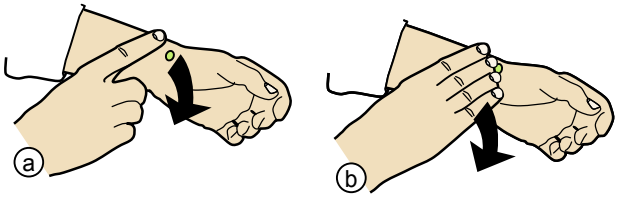

Figure 5: The *small finger problem:* (a) Scanning with a single finger offers only limited range for gestures. (b) This range can be increased by scanning with the entire hand.

PROTOTYPE DEVICES

In order to validate the concept of monolithic devices and scanning we have created simple prototypes of touch scanners (Figure 6) and motion scanners (Figure 7).

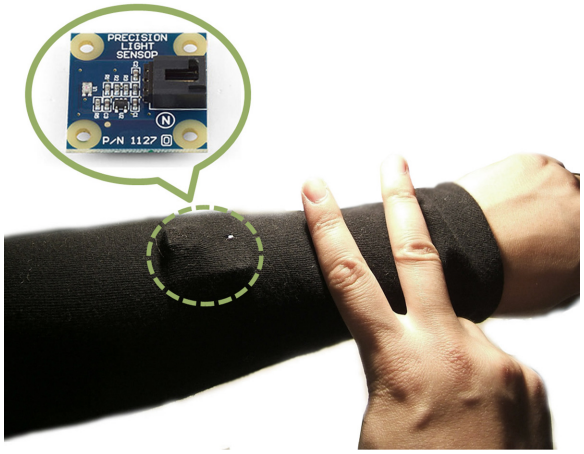

Figure 6: Touch scanner prototype based on a Phidgets light sensor board connected to a PC. To simulate the notion of a disappearing mobile device, we hide the device under the sleeve of a sweatshirt; a 1mm hole exposes the sensor.

All prototypes use a PC for data processing. Input from the motion scanners comes in as pointing input. A mouse hook program intercepts the mouse move events, extracts relative mouse movements as *dx* and *dy*, and delivers them to the application. For the mouse version shown in Figure 7a, input is mirrored left-right to account for the fact that the device is operated up-side-down.

Since none of these devices offers an explicit *out-of-range* state we used a timeout for that purpose, i.e., any period of at least 750ms without *MouseMove* events is considered *out-of-range*.

The laser track point in Figure 7c is a research prototype from the Hardware group at Microsoft, courtesy of John Lutian. It is particularly optimized for tracking human skin.

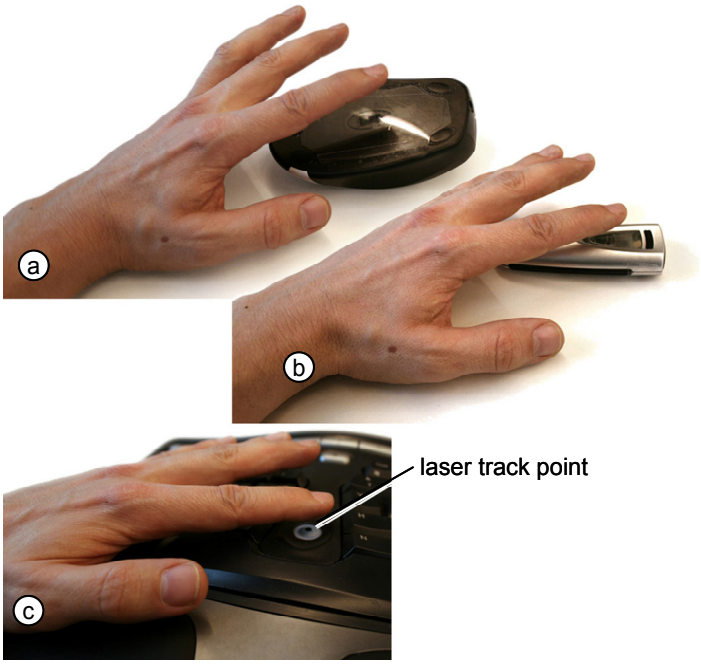

Figure 7: Three prototypes we used to simulate a motion scanner device: (a) Optical mouse on its back, (b) Xybernaut Poma, and (c) laser track point.

Demo Applications

We have used these prototypes to implement a series of simple interactive demo applications running off a PC that illustrate the interaction model.

Air condition control uses on a motion scanner with a single multi-color LED. Double-tapping unlocks the device. The LED responds by displaying its target temperature, such as a shade of blue for cold. Entering "*h*" turns it to heating and makes the LED turn light red. Marking "*up"* increases the temperature one notch at a time; marking *"down"* a few times or entering "*c*" sets it back to cold.

Audio player offers audio output and emulates some of the functionality of the *iPod* click wheel. Marking triggers main functions, including play, pause, and next track. Dialing as shown in Figure 4b browses songs in a playlist. The Graffiti commands "*a*" adds a song to the play-now playlist. Entering "*v*" and dialing adjusts volume. Entering "*s*"

and dialing scrubs within the current track. Modes return on timeout, confirmed with a sound.

USER STUDIES

The interaction techniques we adopted for disappearing mobile devices, such as marking and gesturing, were initially developed for a very different type of user interface hardware. When ported to a disappearing mobile device, these techniques undergo substantial changes. First, as discussed above, users now perform the techniques with the *surface* of the fingertip or hand, rather pen or mouse. Second, due to the occlusion of the device users are likely to commit prematurely as they accidentally gesture over the edge of the device. And third, there is no visual feedback while performing the gesture due to occlusion.

As a result, existing knowledge and performance data about these techniques is unlikely to translate to the new hardware interface and needs to be reestablished in the new context. To begin this process, we conducted a usability test on *marking* adapted to a disappearing mobile device, as well as a user study comparing two types of text entry.

USABILITY STUDY: MARKING ON A DISAPPEARING D.

The purpose of this study was to validate the effectiveness of the marking technique on direction/motion scanner devices and to identify problems and limitations. During each trial, participants entered one mark. Our goal was to determine the reliability of the technique and to identify potential usability issues.

Interface and apparatus

The device used in the study was a simulated *motion scanner*. Because of its high tracking accuracy, we used the simplest of our three prototypes: the optical mouse (*Logitech G5 high precision*, 2000dpi). Participants held the device in their left hand, and entered marks with their right hand's index finger. The device was connected to a PC running Windows XP. The software was written in Visual C#.

Task

During each trial, participants performed a selection from an 8-item marking menu. At the beginning of each trial, a screen showed participants which item to select. To simulate expert performance, we used the descriptive terms "up", "up-right", "right", etc. for menu names. Participants selected the respective item by swiping their index finger in the corresponding direction over the device.

As described above, marking gestures were delimited by a 750ms timeout. Each trial was timed from the moment an item was presented to the participant until the timeout completed. If an incorrect item was selected, an error was recorded. Error trials were not repeated.

Procedure

Before the study, participants were educated about the marking concept. For that purpose, a marking menu with the traditional 8-octant visuals and a cursor trail was shown on a connected computer screen. Users performed 20 practice trials using the device under this visual feedback. After completion of the practice trials, visual feedback was turned off, so all timed trials were performed without feedback. Overall, each participant performed 80 timed trials: 10 for each menu item in randomized order. Overall the study took approximately 15 minutes per participant.

Participants

12 participants (7 male) aged between 26 and 38 were recruited from our local institute. All were right-handed.

Results

On average, trials took 1736ms including the 750ms timeout (*SD* = 289.55ms) to complete. The average error rate across all marking directions was 4.8% (46 errors out of 960 trials combined across all users).

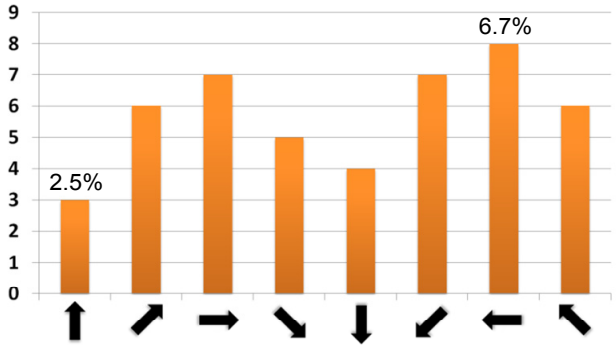

Figure 8: Errors for marks of the respective direction (y-axis is overall number of errors across all users).

Broken down by direction, error rates ranged from 2.5% for "up" to 6.7% for "left" (Figure 8). A one-way Repeated-Measures ANOVA did not find a significant effect of marking direction on error rate ($F_{7, 77}$=.516, p = .82).

Observations about error rates

Observation of participants revealed two factors that caused erroneous selections.

First, errors resulted from a biased orientation of the device. As illustrated by Figure 9a, users would occasionally rotate the device as they were holding it in their non-dominant hands. A correctly aimed mark thereby ended up in a neighboring sector.

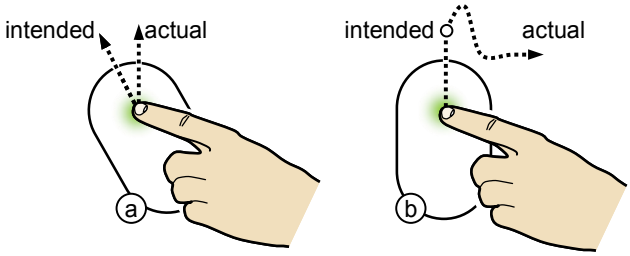

Figure 9: Errors resulted from (a) inadvertent rotation of the device and (b) incomplete lift-off.

Second, errors resulted when participants inadvertently stayed in the range of the sensor after the gesture was presumably complete (Figure 9b). The fact that fingers are thinner than long played a role in this. When held as shown in Figure 9b, participants' fingers cleared the sensor quickly and reliably for "down" and "up" movements. When marking "left" (reversed for left-handed users), in contrast , participants either had to twist their hand in order to get It out of the way or they had to *lift* their hand off the

device in order to get out of the range of the sensor. When participants did not lift their hand far enough off the device, the device continued to track and an error resulted.

This effect that "up" and "down" gestures are easier to perform than others is reinforced by the fact that it is easier to pivot around the elbow than orthogonal to it (see discussion around bottom-left placement of the *Lagoon* in *Alias Sketchbook* [7]).

Discussion

Our usability study provided the following two insights:

Reference system: Traditional marking devices, such as tablet computers, offer a visible reference system; the frame of the screen itself keeps users updated on what physical direction corresponds to "up". The mouse-based prototype used in the study offered only a weak frame of reference; a true disappearing mobile devices will lack such a frame of reference altogether. Our observations, however, suggest that adding a frame of reference can reduce error. One approach is to mount devices at places that are surrounded by features that can serve as landmarks, e.g., the users arm in the case of a wrist-mounted device.

Asymmetries of the finger: not all gestures are equally reliable to perform. When designing a gesture language, easy gestures such as "down" should be used preferably, while hard gestures, such as "left" (mirrored in the case of left-handed use) should be avoided. However, triggering the out-of-range state explicitly using an additional touch sensor can most likely reduce this problem substantially.

Despite these two usability issues, error rates below 5% suggest that marking is a sound input technique for disappearing mobile devices.

USER STUDY: UNISTROKE ON DISAPPEARING DEVICE

The purpose of this user study was to investigate text entry on motion scanners. As discussed earlier, motion scanners afford the entry of sequence of gestures and thus unistroke-based alphabets. In this study, we compared two unistroke alphabets, *Graffiti* and *EdgeWrite*, and evaluated their suitability for use with disappearing mobile devices.

Interfaces

There were two interface conditions.

The *Graffiti* condition was implemented using an open source Graffiti recognizer [15]. Its alphabet differs slightly from the original version by Palm Inc. in that the letters 'B' and 'E' use a lowercase mnemonic, and 'D', 'G', 'P', and 'Q' each use one of the alternate writings also supported by Palm's Graffiti [12]. The alphabet used in the study is reproduced in Figure 10.

Figure 10: The gesture alphabet supported by the graffiti condition.

The *EdgeWrite* condition was implemented using the original EdgeWrite program for mouse and trackballs [47]. Figure 11 shows the supported characters. We configured EdgeWrite with 45-pixel radius, 60° diagonals, and 750ms timeout.

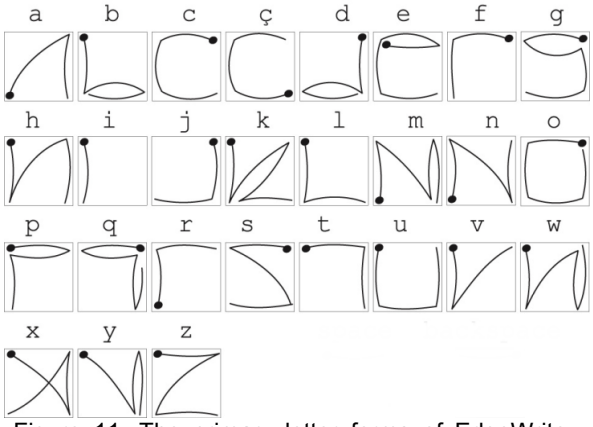

Figure 11: The primary letter forms of EdgeWrite [49, 47]. Alternative forms exist for most characters (not shown).

All interface conditions were run on the same mouse-based device used in the marking study. All input was mirrored left-right to account for the fact that the device that was operated on its back. Neither of the alphabets distinguished between upper and lower case characters.

Unlike the marking study, participants operated the device with their entire hand. In a pilot study, we had asked participants to enter text with a single finger only and found very high (> 40%) error rates. This had clearly resulted from the small-finger problem discussed earlier. Switching to the full hand eliminated the problem.

In addition, participants wore a glove as shown in Figure 12. Piloting had revealed that the ridges in participants' palms impact tracking. This is caused by the fact that the focal range of commercial mice is deliberately cropped in order to minimize tracking errors during clutching. The correct solution is to use an optical sensor without the respective firmware modification. As a fast workaround, the glove solved the problem.

Figure 12: A glove served as a work around to overcome the limited focal range of our device.

Apparatus
We used the same apparatus as in the Marking study.

Participants
A different set of 24 volunteers (19 males) aged between 21 and 41 participated. Three had previous experience with Graffiti on PDAs, but none had used EdgeWrite before. All were right-handed.

Task
Participants entered one character per trial. Participants were presented with the character and then entered it using the respective interface. Graffiti gestures were delimited by a 1500ms timeout; EdgeWrite used a 750ms timeout. Both were tuned carefully in a pilot study to ensure an optimal recognition rate. If the character was not recognized or recognized as a different character, an error was recorded. Error trials were not repeated.

Design
We used a between-subjects design, i.e., 12 randomly assigned participants used the Graffiti interface, while the other 12 used the EdgeWrite interface. The between-subjects design allowed us to keep the study duration reasonably low (40min) by requiring each participant to learn only one of the two alphabets.

Procedure
Before the study, participants received a demonstration and then practiced each character at least 10 times. As in the previous study, visual feedback was provided during training, but not during timed trials. To keep training time within reasonable bounds, a printed copy of the two alphabets (Figure 10 and Figure 11) was visible during timed trials. Training took about 20 minutes on average.

Each participant then performed 8 blocks, during each of which the complete set of 26 alphabetic characters was presented in random order. Finally, participants filled in a questionnaire. Overall, the study took approximately 40 minutes per participants, including training.

We recorded error rate, but not task time. Piloting had revealed substantial differences in learning rates and prior experience and as a result some participants required the sheet while others managed to perform the task without. This impacted task time to a point where we chose not to consider task time in this study.

Hypothesis
Based on the results of comparable studies with trackball [47] and stylus devices [49], we expected the error rate of the EdgeWrite condition to be lower than for the Graffiti condition.

Results
A total of 4992 data points were collected (2496 per technique).

An independent t-test found a significant difference in error rate ($t_{22} = 2.283$, $p = .032$). The error rate of the EdgeWrite condition (5.2%, 131 errors, *SD* 1.35%) was lower than that of the Graffiti condition (6.9%, 172 errors, *SD* 1.92%). This supported our hypothesis.

Subjective ratings of learnability showed the same trend. On a five-item Likert scale (with higher values being better), the learnability of the EdgeWrite condition was rated 3.08 (*SD* = .793), while Graffiti's learnability was rated only 2.42 (*SD* = 1.165). EdgeWrite's usability was rated 3.67 (SD = .888), which is comparable to Graffiti's usability 3.58 (*SD* = .996).

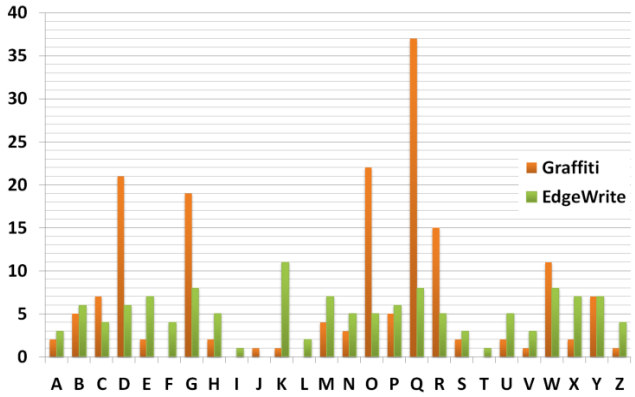

Figure 13: Number of errors (out of 96 entries per technique) recorded for each Roman letter.

Results by character
Figure 13 shows the breakdown of error rate by character. For both interfaces, error rates differed widely between characters. In the graffiti condition error rates were particularly unbalanced, with 'D', 'G', 'O', 'Q', and 'R' together accounting for 66.3% of all errors (114 out of 172).

Figure 14 shows examples of misrecognized characters of these five characters.

Discussion
Following the study, we analyzed our logs to investigate why these Graffiti characters had performed poorly during the study. When the Graffiti recognizer evaluates a stroke, if uses two types of features: (a) the presence of *local* features, such as corners and (b) the *relative position* between stroke segments, such as whether two strokes intersect or not. For the top-five most error-prone characters, recognition relies heavily on such relative position features (Figure 13). Most of the characters with low error rates, in contrast, do *not* rely on relative position features.

On closer inspection, this is not surprising. Both gesture languages were designed for devices that visualize the stroke as it is drawn, which makes it easy to align later stroke segments with segments already on the screen. This type of feedback, however, is missing on disappearing mobile devices, which resulted in increased error rated for gestures relying on this type of feature.

A possible approach to overcoming the lack of visual feedback is to offer a *haptic* reference. On stylus-based devices, the user's palm can serve as such a reference; on a disappearing mobile device, however, the palm is in motion. We found that complementing the device with a small tactile "dot", similar to the tactile dots on the F and J keys of a QWERTY keyboard can serve as such a haptic reference.

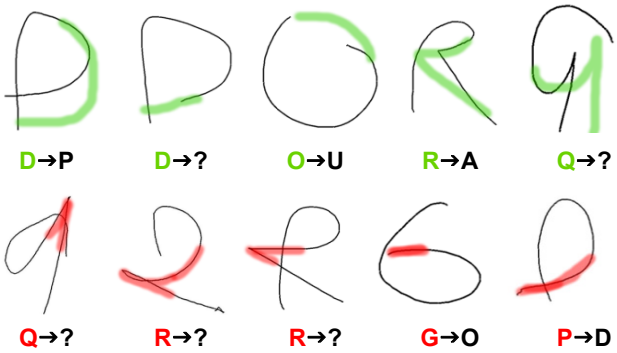

Figure 14: Examples of gestures from the study that were misrecognized because participants (a) undershot or (b) overshot. **D→P** means "D" was presented and the participant's stroke was recognized as **P**. Green/red overdrawn strokes highlight the missing/superfluous part that caused the error.

Summary

The error rate of EdgeWrite was consistent with results reported in [48], indicating that text entry with EdgeWrite on a disappearing mobile device is not only possible, but competitive with larger devices.

The error rate of Graffiti, in contrast, was roughly twice as high as Graffiti run on a pen-based computer (e.g., 3% error rate in [24]). The characters responsible for this performance loss did poorly because they relied on relative position features, which are hard to control on a disappearing mobile device. These findings suggest that gesture languages on disappearing mobile devices need to be (re)designed carefully, so as to be either more forgiving with misaligned features or to instead use gestures that do not rely on relative positions.

The characters from the EdgeWrite condition, in contrast, do not rely on relative positions the way Graffiti characters do. This seems to be a likely explanation for why the EdgeWrite condition performed well on our prototype device.

IMPLICATIONS FOR DESIGN

Based on the two studies presented above we have gained first insights into the characteristics of disappearing mobile devices. Here is a summary of the design implications:

1. Use the entire hand for input to allow for larger motion amplitude. It reduces error with complex gestures.
2. Preferably use marks that do not suffer from the incomplete lift-off problem, such as "down" and "up". Ideally, use a dedicated a touch sensor to implement an explicit out-of-range state.
3. Preferably use unistroke gestures that do *not* rely on the correct recognition of relative position features.
4. If orientation matters, as is the case for marking, provide a clear reference frame, such as physical features located close to the device.
5. To avoid premature commit, provide a tactile feature that tells users when the device is getting near the edge of their hand, e.g., a small tactile dot.
6. Design devices, such that they glance over irregularities and gaps between fingers. However, limit the focal range to prevent motion past lift-off from being recognized as a gesture.

CONCLUSION AND FUTURE WORK

In this paper, we investigated the evolution of mobile devices in one specific direction, namely miniaturization. We investigated what technology are most likely to provide the basis for such devices, what abilities they can be expected to have, and what interaction models can be ported to these devices.

Our main research question was to determine what abilities such devices can be expected to have and whether devices that size can still allow for meaningful interaction. We found that the input capabilities of such disappearing mobile devices are surprisingly comprehensive, allowing for comparably efficient and reliable selection of commands, as well as text entry. Despite their size, the proposed devices do offer an input vocabulary sufficing for a whole range of current consumer devices, such as audio players or potentially even more complex communication devices. The smallness of these devices and their ability to blend into the users' clothing or maybe even their skin will allow for truly ubiquitous use.

As future work, we plan on exploring visual *output* on disappearing mobile devices, such as single-pixel displays. Based on this, we plan on creating self-contained prototypes and studying them in real-world tasks.

ACKNOWLEDGMENTS

We thank John Lutian for providing us with his remarkable laser track point prototype, Jake Wobbrock for sharing the original EdgeWrite code with us, and Ken Hinckley and Gerry Chu for their feedback.

REFERENCES

1. Baudisch, P and Chu, G. (2009). Back-of-device interaction allows creating very small touch devices. In *Proc. CHI'09,* pp. 1923-1932.
2. Brumitt, B., Meyers, B., Robbins, D., Krumm, J., Czerwinski, M., and Shafer, S. (1998). The New EasyLiving Project at Microsoft Research. *Proc. Joint DARPA/NIST Smart Spaces Workshop '98*.
3. Butler, A., Izadi, S., and Hodges, S. (2008). SideSight: multi-"touch" interaction around small devices. *Proc. UIST '08*, pp. 201-204.
4. Cao, X., A. Wilson, R. Balakrishnan, K. Hinckley, S. Hudson. (2008) ShapeTouch: Leveraging Contact Shape on Interactive Surfaces. In *Proc. Tabletop'08,* pp. 139-146.
5. Chi, E. H., Song, J., and Corbin, G. (2004). "Killer App" of wearable computing: wireless force sensing body protectors for martial arts. *Proc. UIST '04*, pp. 277-285.
6. Feiner, S., Macintyre, B., and Seligmann, D. (1993). Knowledge-based augmented reality. *Commun. ACM* 36, 7 (Jul. 1993), pp. 53-62.
7. Fitzmaurice, G., Khan, A., Pieké, R., Buxton, W., Kurtenbach, G. (2003). Tracking menus. *Proc. UIST'03*, pp. 71-79.
8. Fukumoto, M. (2005). A finger-ring shaped wearable handset based on bone-conduction. *Proc. ISWC '05*, pp. 10-13.

9. Fukumoto, M. and Tonomura, Y. (1999). Whisper: a wristwatch style wearable handset. In *Proc. CHI'99,* pp. 112-119.
10. Fukumoto, M. and Tonomura, Y. (1997) Body-coupled FingerRing: wireless wearable keyboard. *Proc. CHI97,* 147-154.
11. Goldberg, D. and Richardson, C. (1993). Touch-typing with a stylus. *Proc. INTERCHI '93*, pp. 80-87.
12. Graffiti. http://en.wikipedia.org/wiki/Graffiti_(Palm_OS)
13. Harrison, C. and Hudson, S. (2008). Scratch input: creating large, inexpensive unpowered and mobile finger input surfaces. *Proc. UIST '08*, pp. 205-208.
14. Hinckley, K., Baudisch, P., Ramos, G., Guimbretiere, F. (2005). Design and analysis of delimiters for selection-action pen gesture phrases in *scriboli*. *Proc. CHI '05*, pp. 451-460.
15. Julien Couvreur's programming blog. Graffiti code download at http://blog.monstuff.com/archives/000012.html
16. Hinckley, K. (2003). Synchronous gestures for multiple persons and computers. Proc. UIST '03. pp. 149-158.
17. Hudson, S. (2004). Using light emitting diode arrays as touch-sensitive input and output devices. *Proc. UIST'04*,pp.287-290.
18. Kurtenbach, G. and Buxton, W. (1994). User learning and performance with marking menus. *Proc. CHI '94*,pp. 258-264.
19. Labrune, JB. And Mackay, W. (2006). Telebeads: social network mnemonics for teenagers. *Proc. Conference on Interaction design and children'06,* pp. 57 - 64.
20. Lawo, M., Herzog, O., Lukowicz, P., and Witt, H. (2008). Using wearable computing solutions in real-world applications. *CHI '08 Extended Abstracts*, pp. 3687-3692.
21. Li, K. A., Baudisch, P., and Hinckley, K. (2008). Blindsight: eyes-free access to mobile phones. *Proc. CHI '08*, 1389-1398.
22. Liess, M., Weijers, G., Heinks, C., van der Horst, A., Rommers, A., Duijve, R., and Mimnagh, G. (2002). A miniaturized multidirectional optical motion sensor and input device based on laser self-mixing. *Measurement Science and Technology,* 13(2002), pp. 2001-2006.
23. Luk, J., Pasquero, J., Little, S., MacLean, K., Levesque, V., and Hayward, V. (2006). A role for haptics in mobile interaction: initial design using a handheld tactile display prototype. *Proc. CHI '06*, pp. 171-180.
24. MacKenzie, I.S. and Zhang S.X. (1997). The immediate usability of graffiti. *Proc. Graphics Interface '97*, pp. 129-137.
25. MacKenzie, I.S. and Tanaka-Ishii, K. (2007). Text entry with a small number of buttons. In MacKenzie, I. S., and Tanaka-Ishii, K. (Eds.) *Text entry systems: Mobility, accessibility, universality*, pp. 105-121. San Francisco, CA: Morgan Kaufmann.
26. Mann, S. (1996). "Smart clothing": wearable multimedia computing and "personal imaging" to restore the balance between people and their intelligent environments. *Proc. Multimedia '96*, pp. 163-174.
27. Matias, E., MacKenzie, I. S., and Buxton, W. (1996). A wearable computer for use in microgravity space and other non-desktop environments. *Proc. CHI '96*, pp. 69-70.
28. Miner, C.S., Chan, D.M., and Campbell, C. (2001). Digital jewelry: wearable technology for everyday life. *CHI '01 Extended Abstracts*, pp. 45-46.
29. Morse code. http://en.wikipedia.org/wiki/Morse_code
30. Moscovich, T. and Hughes, J. F. (2004). Navigating documents with the virtual scroll ring. *Proc. UIST '04*, pp. 57-60.
31. Narayanaswami, C., Kamijoh, N., Raghunath, M., Inoue, T., Cipolla, T., Sanford, J., Schlig, E., Venkiteswaran, S., Guniguntala, D., Kulkarni, V., and Yamazaki, K. (2002). IBM's linux watch: the challenge of miniaturization. *IEEE Computer*, 35(1), January, 2002, pp. 33-41.
32. Olwal, A., Feiner, S., and Heyman, S. (2008). Rubbing and tapping for precise and rapid selection on touch-screen displays. *Proc. CHI '08*, pp. 295-304.
33. Popov, P., Pulov, S., and Pulov, V. (2004). A laser speckle pattern technique for designing an optical computer mouse. *Opt Laser Eng*, 42 (2004), pp. 21-26.
34. 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.
35. Ramos, G., Boulos, M., and Balakrishnan, R. (2004). Pressure widgets. *Proc. CHI '04*, pp. 487-494.
36. Rekimoto, J. (2001). GestureWrist and GesturePad: Unobtrusive Wearable Interaction Devices. Proc. ISWC '01, pp. 21.
37. Russell, D. M., Streitz, N. A., and Winograd, T. (2005). Building disappearing computers. *Commun. ACM* 48, 3 (Mar. 2005), pp. 42-48.
38. Sawant, A.P. and Healey, C.G. (2005). A survey of display device properties and visual acuity for visualization. TR-2005-32, North Carolina State University.
39. Schwesig, C., Poupyrev, I., and Mori, E. (2004). Gummi: a bendable computer. *Proc. CHI '04*, pp. 263-270.
40. Starner, T. (2002). The role of speech input in wearable computing. *Pervasive Computing*, 1(3), pp. 89-93.
41. Starner, T., Auxier, J., and Ashbrook, D. (2000). The gesture pendant: a self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. *Proc. International Symposium on Wearable Computing '00*, pp. 87-94.
42. Vogel, D. and Baudisch, P. (2007). Shift: A Technique for Operating Pen-Based Interfaces Using Touch. *Proc. CHI'07*, pp. 657-666.
43. Weiser, M. (1991). The Computer for the 21st Century. *Scientific American Special Issue on Communications, Computers, and Networks*, September, 1991.
44. Werner, J., Wettach, R., and Hornecker, E. (2008). Unitedpulse: feeling your partner's pulse. *Proc. MobileHCI '08*, pp. 535-538.
45. Wigdor, D., Forlines, C., Baudisch, P., Barnwell, J., and Shen, C. (2007). LucidTouch: a see-through mobile device. *Proc. UIST '07*, pp. 269-278.
46. Wilson, A.D., Cutrell, E. (2005). FlowMouse: a computer vision-based pointing and gesture input device. *Proc. INTERACT'05,* pp. 565-578.
47. Wobbrock, J. and Myers, B. (2006). Trackball text entry for people with motor impairments. *Proc. CHI '06*, pp. 479-488.
48. Wobbrock, J. O. and Myers, B. A. (2005). Gestural text entry on multiple devices. *Proc. Assets '05*, pp. 184-185.
49. Wobbrock, J.O., Myers, B.A., and Kembel, J.A. (2003). EdgeWrite: a stylus-based text entry method designed for high-accuracy and stability of motion. *Proc. UIST '03*, 61-70.
50. Yatani, K. and Truong, K. N. (2007). An evaluation of stylus-based text entry methods on handheld devices in stationary and mobile settings. *Proc. MobileHCI '07*, pp. 487-494.
51. Zhao, S. and Balakrishnan, R. (2004). Simple vs. compound mark hierarchical marking menus. *Proc. UIST '04*, pp. 33-42.
52. Zhao, S., Dragicevic, P., Chignell, M., Balakrishnan, R., and Baudisch, P. (2007). Earpod: eyes-free menu selection using touch input and reactive audio feedback. *Proc. CHI '07*, pp. 1395-1404.