Foreground and Background Interaction with Sensor-Enhanced Mobile Devices

KEN HINCKLEY
Microsoft Research
JEFF PIERCE
Georgia Institute of Technology
and
ERIC HORVITZ and MIKE SINCLAIR
Microsoft Research

Building on Buxton's foreground/background model, we discuss the importance of explicitly considering both foreground interaction and background interaction, as well as transitions between foreground and background, in the design and implementation of sensing techniques for sensor-enhanced mobile devices. Our view is that the *foreground* concerns deliberate user activity where the user is attending to the device, while the *background* is the realm of inattention or split attention, using naturally occurring user activity as an input that allows the device to infer or anticipate user needs. The five questions for sensing systems of Bellotti et al. [2002] proposed as a framework for this special issue, primarily address the foreground, but neglect critical issues with background sensing. To support our perspective, we discuss a variety of foreground and background sensing techniques that we have implemented for sensor-enhanced mobile devices, such as powering on the device when the user picks it up, sensing when the user is holding the device to his ear, automatically switching between portrait and landscape display orientations depending on how the user is holding the device, and scrolling the display using tilt. We also contribute system architecture issues, such as using the foreground/background model to handle cross-talk between multiple sensor-based interaction techniques, and theoretical perspectives, such as a classification of recognition errors based on explicitly considering transitions between the foreground and background. Based on our experiences, we propose design issues and lessons learned for foreground/background sensing systems.

Categories and Subject Descriptors: H.1.2 [**Models and Principles**]: User/Machine Systems—*Human factors*; *human information processing*; B.4.2 [**Input/Output and Data Communications**]: Input/Output Devices—*Channels and controllers*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Input devices and strategies*; *interaction styles*

General Terms: Design, Human Factors

Authors' addresses: K. Hinckley, E. Horvitz, and M. Sinclair, Microsoft Research, One Microsoft Way, Redmond, WA 98052; email: {kenh,horvitz,sinclair}@microsoft.com; J. Pierce, College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30332; email: jpierce@cc.gatech.edu.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 1515 Broadway, New York, NY 10036 USA, fax: +1 (212) 869-0481, or permissions@acm.org.
© 2005 ACM 1073-0616/05/0300-0031 $5.00

ACM Transactions on Computer-Human Interaction, Vol. 12, No. 1, March 2005, Pages 31–52.

SAMSUNG 1025

Additional Key Words and Phrases: Human-computer interaction, context awareness, sensing, augmented devices, input devices, interaction techniques

1. INTRODUCTION

Recent advances in sensor technology have made many inexpensive detectors available [Saffo 1997; Baxter 1997] that can sense the surrounding *context* of the physical environment and human activity. To narrow the gap between the user's understanding and the system's understanding of the current context, we have experimented with a "Sensing PDA" that can detect when the user is holding, tilting, or is proximal to the device, and employ these perceptions to move responsibility for some actions and decisions from the user to the system. The sensors enable the system to sense when the user picks up, puts down, looks at, holds, or walks around with his or her mobile device. These actions represent a hidden vocabulary of *naturally occurring gestures* that people spontaneously exhibit in the handling and use of mobile devices. Our sensing techniques make these types of activities an integral part of interaction with mobile computers.

With mobile devices, keeping interactions minimally demanding of cognitive and visual attention is a core design issue. The demands of real-world activities, such as walking along a busy street or talking to a colleague, may prevent the user from interacting with or paying attention to a device at all. Even a task as simple as tapping a button on the screen steals visual attention from other real-world tasks and can become a burden if the user must focus on external events. Sensors may play a role in addressing such problems by helping to manage the user's attention to foreground activity, or automating some actions in the background so that the user need not directly attend to them at all. But how do our sensors know when to respond? How can they assist the user in transitions from inattention (in the background) to focused attention on the device (in the foreground), or vice versa?

Our experiences lead us to assert that Buxton's foreground/background model is central to reasoning about these types of questions, and to the design of sensor-based interaction in general: sensor-augmented mobile devices should explicitly consider and encode foreground/background states and transitions in their interaction design and in their system architecture. We discuss and explore this thesis in the remainder of this article, which has the following structure. We review the Buxton foreground/background model, define how we interpret the terms "foreground" and "background," and discuss the foreground/background model's relation to the "Five Questions" framework [Bellotti et al. 2002]. In Section 2 we describe the specific sensors used by our system and several interaction techniques that leverage the information provided by these sensors. As we discuss the interaction techniques, we use the issues that arise to illustrate nine general design issues and lessons learned with respect to foreground and background interaction. In Section 3 we present an architecture that our implementation uses to help manage some of these issues. We then discuss in Section 4 some remaining general issues, including analysis of false positive and false negative errors with respect to transitions between

Communication Object	Ground: Foreground	Ground: Background
Human-Human	Telephone Video Conferencing *requires user's direct attention*	"Portholes" system (videoconferencing with background sensing of participant availability; Dourish & Bly 1992) *user's attention not required at all times*
Human-Computer	GUIs Sensing PDA (tilt-to-scroll) *user's attention is on interface* *user issues explicit commands*	Smart house technology Sensing PDA (automatically changing the display format) *user's attention not required at all times* *actions taken on behalf of user*

Fig. 1. Buxton's model of technology-mediated communication and interaction based on the communication object (human or computer) and the ground (foreground or background), with examples of each class and some distinguishing characteristics. Adapted from Buxton [1995].

foreground and background and extensions to the Five Questions framework. We also consider the question of mapping sensed user activity to inferred system actions, and potential mechanisms to allow the user to control or configure such mappings. Section 5 summarizes the main contributions of the article and suggests some areas for future work.

1.1 Models of Foreground versus Background Interaction

Buxton proposes a classification of Human-Computer Interaction into two broad categories: foreground interaction and background interaction [Buxton 1995], as seen in Figure 1. Buxton defines *foreground interaction* as "activities which are in the fore of human consciousness—intentional activities." The foreground is characterized by explicit step-by-step human guidance for the computer; for example, in the classic direct manipulation metaphor of graphical user interfaces, the user must focus his or her direct attention on the interface to make progress. An example of a sensor-based foreground interaction that we explore is scrolling the display of a small device as a function of tilting of the device. In our view, the distinguishing characteristic of the foreground is that it requires the user's direct attention, as shown in the left column of Figure 1.

By contrast, Buxton [1995] defines *background interaction* as "tasks that take place in the periphery–'behind' those in the foreground." What Buxton calls the background closely relates to other terms used in the literature, including "noncommand" interaction [Nielsen 1993], "implicit" interaction [Harrison et al. 1998], and "incidental" interaction [Dix 2002]. Supporting background interaction requires methods for sensing the user's activity and inferring appropriate actions. We define *background sensing* as sensing an action that the user would have had to perform anyway to accomplish their task (i.e. the action is not an artificially contrived command that exists solely for the purpose of

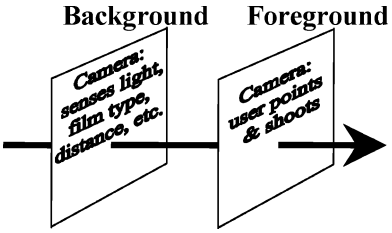

Fig. 2. Foreground and background interaction working together in simultaneous, parallel layers (adapted from Buxton [1995]), with the point-and-shoot camera as a familiar example.

communicating user intent to the computer). Many of our sensing techniques, such as switching between portrait and landscape display modes when user holds the device in a desired orientation, fit the criteria of background sensing techniques. In this example, the user has to rotate the display to see it at the new orientation anyway; detecting this and automatically changing the display format pushes the interaction into the background.

Background sensing can also operate *in support of* foreground activities that require the user's direct attention. Here, foreground/background interaction characterizes two parallel layers of interaction that support one another (Figure 2). Buxton describes the familiar example of the point-and-shoot camera: the operator is responsible for the foreground activities of choosing what to photograph (point) and when to photograph it (shoot); this simple operation is made possible by a plethora of sensors that capture information in the background about the ambient light level, composition and distance to the subject, the type of film loaded, and so forth.

Ishii and Ullmer [1997] offer a different view of foreground and background: they discuss graspable objects as occupying the foreground, while ambient displays of information linger in the background. Since our sensor-augmented device *is* a graspable object, a strict interpretation of the view that Ishii and Ullmer present would suggest that all of our research addresses the foreground. But just because Ishii and Ullmer's tangible interactions represent foreground techniques, this does not necessarily mean that *all* device-centric interactions belong to the foreground. Buxton's example of the point-and-shoot camera clearly makes this point: the sensors on this device push many aspects of configuring the camera into the background, behind the fore of the user's attention.

We can reconcile Ishii and Ullmer's interpretation of foreground and background interaction with Buxton's (and our own) view if we classify foreground and background not in terms of device-centric versus environmentally situated sensors and displays, but rather in terms of *the degree of attention that an interaction requires*. Using attention as our yardstick also allows the foreground/background distinction to become a continuum rather than a strict dichotomy. For example, we have explored holding a mobile device to one's ear as a gesture to automatically trigger recording a voice memo. This gesture falls into a "middle ground" between background and foreground because holding an object to one's ear is an intentional act, yet by using sensing techniques to

push the mechanical details of activating the recording feature into the background, we simplify activation of the feature and may demand less of the user's attention.

1.2 Foreground/Background Interaction and the Five Questions Framework

One proposed framework for thinking about the issues raised by sensing techniques, the "five questions for sensing systems" derived from Bellotti et al.'s [2002] analysis of human-human communication, primarily considers explicit commands and communication. For example, how does a user know the system is attending to a request? How does the system know what object the user's command (e.g. Save) relates to? In the foreground/background view of interaction, all such explicit communication occurs in the *foreground*. Since their approach is founded upon intentional human-human communication, their framework raises questions for foreground interaction, but does not explicitly address background interaction. However, the work does make important contributions to consideration of user errors (How do I avoid mistakes?) or system errors, which could occur in either foreground or background interaction. Our research extends this perspective to show how consideration of errors is important for foreground and background interaction, as well as transitions between foreground and background.

A critical issue for sensor-augmented devices is to consider the boundaries between foreground and background. Our work with sensor-enhanced devices has led us to conclude that symbiotic background/foreground interaction raises new questions and issues beyond those considered by the five questions framework. Designers of sensing systems must carefully deliberate about transitions between the grounds, both at design time and in real-time as the system handles sensor data. Our Sensing PDA detects and deals with these transitions, allowing our software to reason about when the device is being brought into the foreground, or drifting into the background. Designers must carefully consider the potential for introducing unwanted transitions as the result of false positive failures (when sensor data is mistakenly recognized as a gesture that did not occur), as well as missing transitions as the result of false negative failures (when sensor data contains a gesture that did occur, but the system does not recognize it). Our research suggests examples of user tasks and activities that sensing techniques can support, showing how we can combine multiple sensors to support one interaction, while also using one sensor to support multiple interactions in both the foreground and background, without introducing unwanted cross-talk between techniques.

2. INTERACTION TECHNIQUES, DESIGN ISSUES, AND LESSONS LEARNED

Here, we discuss the following sensor-based techniques that span the continuum from foreground manipulation to automatic or semi-automatic background services:

Automatic Power Control: The device turns itself on when the user picks it up;

Physical Sensor	Context Attribute	Description
Touch Sensors	`Holding`	True if the user is touching any part of the back and/or sides of the device.
	`TouchingBezel`	True if the user is touching a second touch sensor on the left edge of the screen bezel.
	The touch sensors are areas of conductive material on the surface of the device. They detect bare skin contact via capacitance sensing.	
Tilt Sensor	`TiltAngleLr` `TiltAngleFb`	Left/right and forward/back tilt angles of the device's screen relative to gravity
	`TiltOrientation`	Screen orientation corresponding to the direction that the device is tilted: *Portrait*, *LandscapeLeft*, *LandscapeRight*, *PortraitUpsideDown,* or *Flat* (not tilted).
	`LookingAt`	True if the user is likely to be looking at the display, based on expected viewing angles for the current screen orientation.
	`Moving`	True if the device is moving significantly (very slow or slight motions do not trigger this state).
	This is a two-axis linear accelerometer. We interpret its output as a pair of tilt angles, but other accelerations such as vibrations or movement may influence the sensor.	
Gravity Switch	`GravityDirection`	*RightSideUp, UpsideDown*
	The tilt sensor cannot distinguish up from down, so we use a gravity switch to do this.	
Proximity Sensor	`Proximity`	Estimated distance in cm to proximal object.
	`ProximityState`	`Proximity` classified as *Close*, *InRange*, or *OutOfRange*
	The proximity sensor is an infrared sensor/emitter pair that senses the distance to objects within 25 centimeters of the screen. The sensor reading is influenced by the ambient light level and the reflectivity of the object.	

Fig. 3. Sensors used by our system and some context attributes available to applications.

Listening Detection: Holding the device like a phone records voice memos;

Portrait/Landscape Switching: The device changes the display format when the user holds it at a new orientation;

TiltScroll: The user can tilt the device to scroll the contents of the display.

In this discussion we illustrate a number of design issues and lessons that we learned that focus on handling foreground/background states in sensing systems, as well as the transitions between these states. Although we discuss the main design issues and interactive behavior of each technique, for more complete implementation details and further discussion of user reactions and feedback consult Hinckley et al. [2000].

2.1 Sensor Hardware and Derived Context Attributes

We focused our research on sensors that are small, inexpensive, and have potentially very low power consumption, making them practical candidates for integration with mass-produced mobile devices. We use a Compaq iPaq for our current prototype. Figure 3 summarizes the sensors our system uses and software inferences (logical states that we call *context attributes*) it derives from the sensor data.

The context attributes described in Figure 3 form the building blocks of our interaction techniques, and as such provide a convenient shorthand to describe

the techniques. Henceforth in this article, names in the `Courier` font represent context attributes, while *`Italicized`* items represent particular named values of a context attribute.

2.2 Automatic Power Control

Finding the power button on mobile devices can be physically awkward and demanding of attention. We observed that people typically hold their PDA in a posture appropriate for use, and *then* press the power button; pressing the power button is an explicit step secondary to the natural act of picking up the device.

By sensing when the user picks up the device to use it, we push the activation of the device into the background. Automatic Power-On is an example of sensing a transition in the user's attention from the background to the foreground with respect to the device. When a user picks up and looks at the device, we assume the user intends to employ the device in the foreground of attention. However, we classify sensing and responding to this gesture as a *background sensing* technique because it moves the responsibility for activating the power from the user to the system; the user would otherwise have to first attend to the mechanics of finding and pressing the power button. Furthermore, picking up the device is a naturally occurring gesture that is already part of using the device, but represents an action that has not been sensed by traditional systems. To underscore this, all test users who have tried the system have discovered this capability without any instruction; as soon as one picks up the device to try to use it, the device turns itself on. A related technique uses touch sensors to activate an application on a device [Schmidt 2000], but does not sense when the user picks up the device for active use.

We detect when the user is ready to use the device as follows: when the power is off, if the user is `Holding` the device and `LookingAt` the display in *`Portrait`* orientation (but not *`Flat`*), while the `GravityDirection` is `RightSideUp`, and this state persists for a short timeout, then our sensor system powers up the device. This results in the following high-level behaviors:

1. The device cannot power up when in the user's pocket or purse because the user is not holding it.
2. The device will not turn on if the user simply touches it or pushes it out of the way while it is resting on a desk (resting *`Flat`*). The user must be holding the device at tilt angles consistent with viewing the display.
3. The short timeout prevents the device from powering up accidentally due to transient signals, but is short enough that the user does not feel as if he or she has to wait for the action to occur.
4. The device typically will not turn on if the user handles it, but does not look at it (e.g. grabbing the device, but then holding it at one's side rather than using it).

In our own use of the device, we have observed that it occasionally turns on by accident when it is handled. However, when we had test users try the technique, users were not bothered even if the device did power on by accident:

for example, one user commented that he "didn't care" because he would put the device away if he wasn't using it. This suggests that for this technique, false positive recognition has few negative consequences: if the device turns itself on, this is not demanding of the user's attention; the main negative consequence seems to be that some power will be consumed unnecessarily. Thus we bias the algorithm slightly towards eager recognition, as a false negative failure (or even a short delay) when recognizing the gesture could have worse consequences. Picking up the device and looking at it, only to have it remain off in some cases, would force the user to switch his attention to determining why the device was off, and deciding how to activate the power, instead of moving directly to his foreground task.

We considered a corresponding automatic power-down feature (when the user just holds the device at his side, or puts it down on a table, for example) but the user benefit seemed dubious. At best, the user would not notice the device shut down, and a small amount of power might be saved. At worst, false-positives could annoy the user: for example, the user may want to put down the device to use it on a table; if the device powered off every time the user put it down, it would interfere with this manner of use. This would represent a system-initiated transition from the foreground (active use of the device) to the background when in fact the user was attending to the device in the foreground state, without any intention to stop using it.

Preventing the device from powering off at inappropriate times does seem to have value, however. Mobile devices use "inactivity timers" that power the device off or dim the screen if no stylus or button press events have occurred recently. However, these timers can be vexing when one is trying to read the screen, or think about something in the course of a task. Our Sensing PDA infers continued user activity based on a combination of the touch, proximity, and tilt sensors, and resets the system inactivity timer on the user's behalf to prevent these unnecessary interruptions.

2.3 Design Issues and Lessons Learned for Automatic Power Control

Here we summarize some of the high-level insights that this interaction technique led us to, which we formulate as preliminary design principles for foreground/background sensing systems. We give these "lessons learned" sequential numbers for later reference.

L1. *Use background sensing to assist users in transitions to the foreground.* If a sensing system correctly senses that the user has changed the focus of their attention, and is moving from the background to the foreground with respect to the system, this may represent an ideal time to provide an automatically inferred service, such as powering on the device. In this case, as much activity as possible should be handled by the system in the background, to free the user from having to attend to unnecessary steps or details in the foreground.

L2. *Preserve the user's focus of attention by minimizing disruptions to the ground*. The flip side of L1 is that systems should strive to preserve the user's focus of attention; if a sensing system *incorrectly* infers a background to foreground transition, or vice versa, this disrupts the ground. Such an incorrect

inference may cause an untimely interruption that requires users to attend to some aspect of the system (i.e. transition to the foreground) when they otherwise would not have had to do so. Our desire to avoid this kind of problem led us to abandon ideas to develop an automatic feature to power off the system. In general, our system strives to minimize false-positive background to foreground transitions by using sensors to provide multiple points of evidence about the user's activity, short timeouts to make sure that new sensor states continue to persist, and generally conservative policies when providing automated services (e.g., if the device does power on by accident, users feel that this has little or no significant negative consequence; but powering off by accident could significantly interrupt the user).

2.4 Listening Detection

Many current PDA devices include voice recording features, but finding the right button or activating a control on the screen can require significant visual attention. We allow the user to record a voice memo by simply holding the Sensing PDA like a cell phone and speaking into the device—a familiar gesture that requires little or no direct visual attention. The user's impression is that he or she just speaks into the device to start recording. This technique seemingly has elements of both foreground and background interaction, as listening to a device is clearly an intentional foreground activity, yet by eliminating the need to find and press buttons, the technique pushes some aspects of the interaction into the background. We have also experimented with Listening Detection as a way to answer incoming phone calls or switch between speakerphone and private use modes on a device that supports telephony features [Hinckley and Horvitz 2001]. It is possible that users could want to employ this gesture to answer calls, record voice memos, switch to speakerphone mode, or even cue a voice recognition mode, but is not clear that one device could support all of these options simultaneously. We return to this issue in Section 4.2 of the discussion where we discuss *mappings* of recognized activities to automatic services.

Our implementation of voice memo detection combines all of our sensors. First, the user must be holding the device; this prevents accidental recording when the device is in a purse or briefcase. Second, the user must hold the device in Close proximity (within approximately 3 cm), to start speaking into it. Finally, the user must tilt the device within a range typical for holding a telephone handset; that is, we sense the natural posture that the hand makes when holding the device to one's ear.

If these conditions hold true for a short timeout, the device makes a distinct click (to give early feedback that the gesture has been recognized), and then starts the device's voice recording software. This software issues a single sharp beep just before it starts recording, after which the user can leave a voice memo. When they are finished speaking, users naturally move the device away, which automatically stops the recording. The software issues two sharp beeps when recording stops. All of this audio feedback seems crucial to the interaction, as it provides nonvisual feedback of the gesture recognition, cues the user when to start speaking, and confirms that the memo was recorded.

The main usability problem with the technique is that users do not expect devices to be able to react in this way, so in our experience, test users do not discover the gesture unless prompted. However, we find that telling users to "talk into it like you would talk into a cell phone" is the only hint that is necessary for them to use it successfully. Users who have tried the technique did not feel that the gesture was any faster than pressing a button, but reported that it seemed to require less concentration: "I have to think about finding the button, pushing it, holding it," but "with the [sensors] it was just listen for the beep." The sensed gesture seems to better support the user goal of recording a message without extra steps to concentrate on or remember, and a preliminary dual-task experiment suggested that it may require less of the user's attention than the device's manual technique for activating the same feature [Hinckley et al. 2000].

2.5 Design Issues and Lessons Learned for Listening Detection

L3. *Provide feedback of transitions between the grounds and awareness of whether user activity will be interpreted as foreground or background*. For example, our Listening Detection technique uses audio cues to inform the user when the device has recognized that they are listening to it, cueing the user to start talking. Muscular tension and the pose of the hand serves as a constant but subconscious reminder that the voice recording "mode" is still active [Sellen et al. 1992], and all that is required to return the device to the background is to relax one's arm or put down the device.

L4. *Scope foreground interpretation of possible interactions via background sensing*. Systems can use background sensing to constrain the space of possible interactions that must be considered, allowing the system to present a few alternatives for the user to choose from (e.g. proximal selection [Schilit et al. 1994]) or automatically execute an action if it is the most likely or only possible operation (e.g. auto power-on when the user picks up the device). When a specific context such as listening to a device like a phone is recognized, this approach can also allow a general-purpose device with many capabilities to act as an appliance-like device with a specific use in a given context.

2.6 Portrait/Landscape Switching

Unlike a stationary desktop monitor, users of mobile devices can move their displays to look at them from any orientation. When the user holds a mobile device, he will naturally tilt it so that the screen faces himself; this is a movement exhibited naturally by users without any prompting. Using the tilt sensor, we detect these movements and automatically reformat the display to suit the current viewing orientation. For example, a user reading an electronic book or inspecting a spreadsheet may find a portrait or landscape display format more pleasant depending on the document content (Figure 4). Although further details of our algorithm appear in Hinckley et al. [2000], in this section we describe a slight improvement using the `Moving` state.

To view the screen at a new orientation, the user has to move the device. We sense this activity in the background and use it to automatically change the

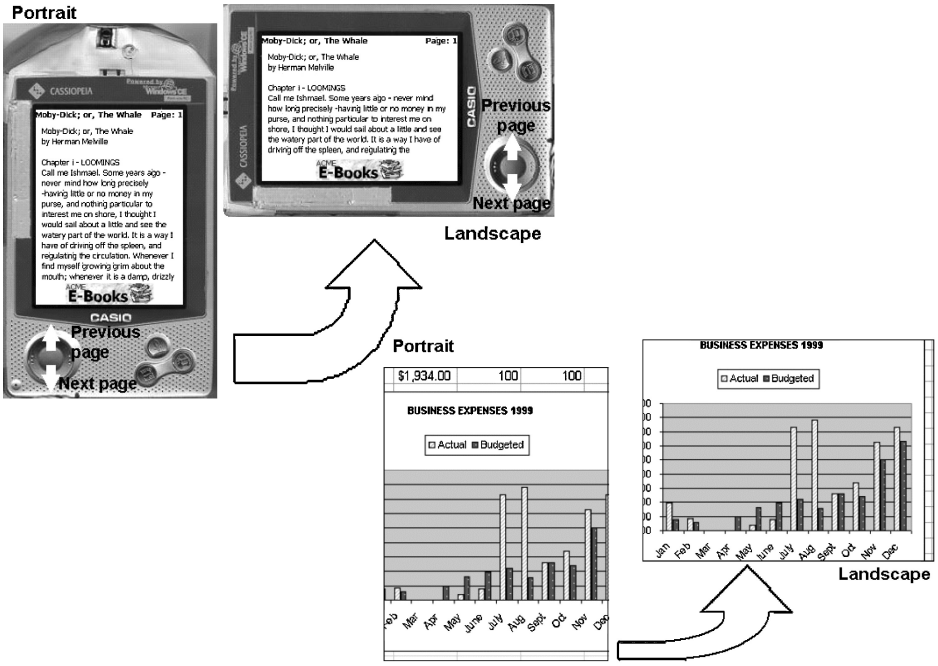

Fig. 4. Portrait/Landscape display mode detection. **Top Left**: An E-book application automatically rotates and reformats the UI to fit the new screen orientation. **Bottom Right**: Spreadsheet application. The user can get the most out of a small display.

display format. Although holding the device in a different orientation represents an intentional act by the user, we classify Portrait/Landscape switching as a background sensing technique. It occurs in the context of continuous use of the device in the foreground, but since the user must move the device to see it in the new orientation, the technique pushes the transaction cost of switching display modes from explicit user control to automatic system inference in the background.

One case that results in some design difficulties occurs when the device's `TiltOrientation` enters the *`Flat`* state. This can occur when the user puts down the device, or uses the device while resting it flat in his lap. For example, with our initial designs, we found that while putting down the device, the user may inadvertently tilt it and change the display format, forcing the user to sometimes pick up the device again, change the orientation, and put it down more carefully. To address this problem, our current design uses the tilt sensors to monitor whether the device is moving significantly. The system does not allow the display format to change until a short timeout after the device stops `Moving`. When movement stops, if the device is resting flat, no change takes place. Likewise, this prevents the screen from changing formats if it is tilted slightly while it is simply resting in the user's lap; a distinct movement to a new orientation is necessary to trigger a display format switch.

Several test users commented that they could easily show information on the screen to a friend or co-worker seated across a table by simply tilting the

display towards that person. The technology affords such quick, informal sharing of the display because it responds quickly, has minimal overhead, and does not interrupt the flow of the conversation. However, one test user did express concern that the display might change orientations if she twisted it while showing it to someone seated next to her: "I think it would drive me nuts. . . I liked it better when I had control of it."

In related work, Schmidt uses a set of mercury switches to adapt "the screen orientation [. . .] to device orientation whenever a stable change in orientation is sensed" [Schmidt et al. 1999]. Bartlett switches display modes if the user stands the device on edge for about 2 seconds [Bartlett 2000]. We quickly determine the orientation based on the naturally occurring activity of the user, rather than relying on a contrived gesture to switch display formats.

2.7 Design Issues and Lessons Learned for Portrait/Landscape Switching

L5. *Automate blocking steps in the background by sensing and removing unnecessary barriers to interaction*. If continued use of the device requires or could greatly benefit from a change in some aspect of the device configuration, this becomes a *blocking step* that must be completed before the user can continue effective interaction. With traditional techniques, the user would have to find and identify the correct hardware button or control panel in the software in order to change the screen orientation, and then they could reorient the device. Our sensing technique automates this blocking step so that the user may simply hold the device as desired. Automatic Power Control also provides an example of removing a blocking step: the user cannot actively use the device in the foreground without turning on the power.

L6. *Include ground selection mechanisms that help indicate whether activity should be interpreted as foreground or background*. Sensing systems must include "selection" mechanisms to decide if user activity should be interpreted as foreground (an intentional act by the user) or background (incidental movement or activity that should remain fully backgrounded). Sensing systems may be able take advantage of subtle, implicit cues to infer user intentionality. Motion sensing is one such implicit ground selection mechanism: Portrait/Landscape switching uses the `Moving` state to sense when the user has completed a distinct movement to a new orientation. This avoids accidental changes to the orientation where the user may inadvertently tilt the display in the course of other actions, such as putting down the device. Automatic Power Control provides another example: it is not possible to power on the device unless the user is `Holding` the device.

Further important issues and lessons learned for Portrait/Landscape Switching arise when this background technique is combined with the TiltScroll foreground sensing technique, and are discussed as L7, L8, and L9 at the end of the following section.

2.8 TiltScroll

Several researchers have explored using tilt to scroll the display of mobile devices [Bartlett 2000; Harrison et al. 1998; Rekimoto 1996; Small and Ishii

1997]. We felt it was important to experiment with this technique, which we call "TiltScroll," as it represents a foreground interaction technique that raises potentially different issues from the background sensing techniques we have discussed thus far. TiltScroll is representative of a number of foreground interaction techniques proposed for mobile devices augmented with tilt sensors. For example, tilting has been used for menu selection and map navigation [Rekimoto 1996], browsing through contacts [Harrison et al. 1998], and text entry [Partridge et al. 2002; Wigdor and Balakrishnan 2003].

TiltScroll allows two-dimensional movement through a document. It maps tilt angle to scrolling speed using a rate-control function (see Hinckley et al. [2000] for details); the user must maintain visual attention while scrolling. Since TiltScroll is explicitly initiated by the user, continuously controlled by the user, and requires the constant visual attention of the user, it is clearly an example of a foreground interaction technique.

We use contact with the screen bezel (`TouchingBezel`) to initiate scrolling. Scrolling continues until contact stops. An advantage of using the bezel touch sensor to engage scrolling is that the sensor has a large surface area and does not require muscle tension to maintain contact. However, we found that users sometimes touch it by mistake, particularly when holding the device in the other display orientations; providing four touch sensors, one for each edge of the screen bezel (but ignoring three of them, depending on the display orientation) may offer a solution to this difficulty.

The Rock'n'Scroll device [Bartlett 2000] uses scrolling as the default mode, allowing scrolling without any explicit clutching mechanism to provide "start" and "stop" signals. To freeze the screen, Rock'n'Scroll uses a tilt gesture that locks the display; this requires the user to disturb the scrolling position to execute the gesture, although the system retroactively attempts to "undo" any such inadvertent scrolling. Several systems set a predefined or user-selectable "zero orientation" relative to which scrolling takes place [Bartlett 2000; Harrison et al. 1998]. Our system instead uses the device's orientation when the user initiates scrolling, allowing scrolling in any display mode and almost any comfortable posture.

Integrating TiltScroll and Portrait/Landscape Switching presents some difficulties. The techniques suffer from a *cross-talk problem* because tilting scrolls the display, yet tilting also may result in changes to the display format. Obviously, the screen should not change orientations during tilt-scrolling. Also, when the user stops scrolling (by releasing the screen bezel), the screen may be tilted towards a new display orientation. The screen should not immediately change display modes at this point, as it would be disconcerting.

2.9 Design Issues and Lessons Learned for TiltScroll

L7. *Prefer background interpretation for typical events*. In our system design, we prefer background interpretation for typical events. Background channels should be kept free for sensing of typical, naturally occurring gestures. Foreground interaction should be reserved for atypical special cases where the user wishes to explicitly control something. For example, because Bartlett makes

tilt-to-scroll the default mode of his tilt-sensitive handheld device [Bartlett 2000], his system sacrifices the ability to sense naturally occurring gestures. His system requires an explicit, contrived gesture to get out of the scrolling mode and have the system observe how the user is naturally holding the device; a foreground interpretation of tilting (tilt-to-scroll) is always active unless the user commands otherwise. By contrast our system exhibits background preference, and thus enables a number of background sensing techniques, including Automatic Power Control and Portrait/Landscape Switching. Yet our device can also support foreground interactions such as TiltScroll by giving the user mechanisms to explicitly indicate and control foreground interpretation.

L8. *Provide explicit ground selection mechanisms that allow foreground techniques to dominate background interpretations*. Although systems may be able to leverage implicit ground selection mechanisms (L4), this is not always possible or even desirable. To maintain user control and a deterministic response for foreground techniques, the system must respect the intention of the user and make it easy to provide an explicit signal to transition to foreground interaction. This also implies that the foreground should heavily dominate or exclude background interpretations of an activity during such explicit signals. Our system achieves this by masking background interpretations of a sensor when the user explicitly indicates a desire to initiate foreground manipulation (e.g. touching the screen bezel to initiate TiltScroll).

L9. *Explicitly encode ground transitions*. Sensing systems should recognize and explicitly encode transitions between foreground and background and use such encodings to minimize disruptive changes to the current mode of interaction. For example, our experiences with integrating the foreground TiltScroll technique and the background Portrait/Landscape Switching technique suggest that a mechanism is needed to prevent cross-talk between the techniques. Our system addresses this cross-talk problem by implementing a new event architecture that provides policies and mechanisms for applications to request and monitor transitions between the grounds, as discussed in the following section.

3. ARCHITECTURE FOR BACKGROUND/FOREGROUND TRANSITIONS

In our experience, designing methods for handling dependencies, transitions, and conflicts among background and foreground modalities is an important design challenge. Sensing systems need architectures and policies that can effectively resolve cross-talk between multiple techniques. Here, we propose an architecture that embodies the Buxton foreground/background model of interaction in its sensor handling policies, thus providing a principled mechanism to support sharing of sensors across multiple applications.

Our system architecture is based on a client/server arrangement with all sensor notifications occurring via message passing. The client uses the *RequestNotification* call to query a context attribute. The Context Server replies immediately with the current value, and then sends subsequent messages whenever the sensor changes value (Figure 5).

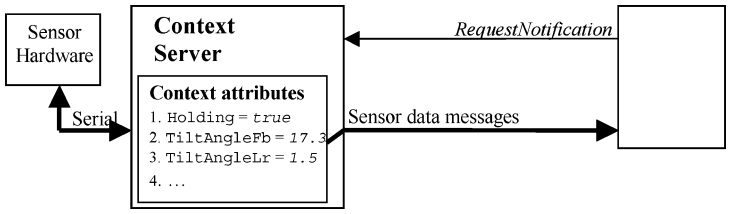

Fig. 5. Block diagram of the client/server architecture.

An application must explicitly characterize the *ground* of how it intends to use a context attribute when it calls *RequestNotification*. We define four levels of ground: *ForegroundService*, *BackgroundService*, *BackgroundMonitor*, and *StopUsing*. These levels embody the machine's view of the Buxton foreground/background model and leverage it to help handle sensor cross-talk between applications.

When an application such as TiltScroll requests a *ForegroundService*, it indicates that it will use the sensor to directly manipulate elements of the interface through foreground interaction. By contrast, if an application wishes to provide a sensor service in the background, which looks for an anticipated pattern of user activity, it requests a ground level of *BackgroundService*. Two additional ground levels round out our mechanism. If an application monitors a sensor to keep a record or log, but does not directly provide a service that may demand the attention of the user in real-time, it can request a ground level of *BackgroundMonitor*, allowing it to observe sensor data in the "deep background," independent of background/foreground transitions. Finally, applications can also halt messages by requesting to *StopUsing* a sensor.

To support transitions between these levels, the Context Server provides a *GroundChange* message to notify applications when they must *StopUsing* a sensor, or when they can resume using a sensor at the application's previously requested level of service. Applications can also take the initiative and request a change to their ground level by calling *RequestGroundChange*. If the request is granted, the Context Server sends out *GroundChange* messages to inform other applications of the change.

Of course, users can explicitly bring an application to the foreground by switching to or starting that application. The Context Server determines which application is in the foreground by monitoring the system input focus; our architecture then routes sensor updates (notification messages) for foreground interaction techniques to the focus application, while withholding updates from background applications that use the same sensor. The Context Server sends a *GroundChange* message to an application telling it to *StopUsing* a sensor if either of the following criteria is met:

1. The application is providing a *BackgroundService* using a sensor for which the active application initiates a *ForegroundService*;
2. The foreground application is providing a *ForegroundService* and it becomes inactive (e.g. the user switches to another application)

Conversely, the Context Server sends a *GroundChange* message to an application telling it to resume using a sensor at its previously requested ground level under the following conditions:

1. The application requested *ForegroundService* and it becomes the active application (e.g. when the user switches to that application).
2. The application requested *BackgroundService* and an application with *ForegroundService* on the same sensor becomes inactive.
3. The application requested *BackgroundService* and the active application explicitly relinquishes *ForegroundService* using the *RequestGroundChange* function.

Using these mechanisms, it is possible to integrate Portrait/Landscape Switching with TiltScroll in an elegant fashion. For example, if the user touches the screen bezel to initiate scrolling in an application that supports it, the application initiates a *ForegroundService* using the tilt sensor. It then receives continuous updates for the tilt angles. Meanwhile, our Portrait/Landscape Switching application receives a *GroundChange* message telling it to *StopUsing* the tilt sensor, so it cancels any pending screen orientation changes. However, even if the Portrait/Landscape Switching application were to ignore the *GroundChange* message, it stops receiving messages from the Context Server. Hence *GroundChange* messages provide an opportunity for applications to interact smoothly with others, but no further explicit programmatic action is required.

When the user stops touching the bezel, the scrolling application requests to *StopUsing* the tilt sensor via the *RequestGroundChange* function. The Context Server then sends the Portrait/Landscape Switching application a *GroundChange* message telling it to resume its *BackgroundService*. When the Portrait/Landscape Switching application receives this message, it clears its state, and it starts to receive new messages updating the tilt angles. Thus our cross-talk mechanism allows the system to shield the Portrait/Landscape Switching application from mistakenly interpreting manipulative tilting of the display resulting from TiltScroll.

Our mechanism provides a foundation that can help developers author responsible applications that effectively share the sensor data without interfering with other (possibly unknown) uses for the same sensors. The resulting place-holders for message handling in the application source code also remind application developers to provide graceful transitions in response to these messages. However, our mechanism does not stop malicious programs from improperly using the sensor data, nor does it eliminate inherent conflicts or prevent race conditions between multiple recognizers that might respond to similar patterns of activity. These problems are important, but seem independent of foreground/background interaction issues.

4. DISCUSSION

4.1 Recognition Errors in Light of Foreground/Background Transitions

Much of the discussion in Section 2 revolves around our efforts to carefully consider potential sensor errors, failure modes, and designs that mitigate or

Ground Transition	Error Type: False Positive	Error Type: False Negative
Back → Fore	(Type I) Background becomes foreground when it should not. *Incidental handling of device interpreted as record memo gesture.* *Screen changes display orientation while the device is resting in the user's lap: The user must deal with a display that is now at the wrong orientation.*	(Type II) Background fails to become foreground when it should. *User holds device like a phone but the gesture is not recognized.* *User keeps holding screen at a new orientation after scrolling, but the display format does not change because it was not notified of orientation changes during scrolling.*
Fore → Back	(Type III) Foreground manipulation incorrectly interpreted as background activity. *User listens to device because it is making a noise; software interprets this as desire to record memo.* *User tips tablet to avoid glare on screen, and display changes format.*	(Type IV) Foreground burdens user with tasks that could be automated in the background. *User speaks into device like a phone but traditional hardware/software cannot recognize this.* *User explicitly switches display format: could have been automated if user knew the sensors support this.*

Fig. 6. Classification of recognition errors by failure mode (false positive or false negative) and the type of transition (background to foreground, or foreground to background).

prevent various sources of erroneous recognition. Here, we contribute a new classification of errors in sensing systems that considers both false positive and false negative error conditions in light of transitions between the foreground and background. This leads to four general classes of errors, Type I through Type IV, as shown in Figure 6. This matrix enumerates potential errors for these four basic classes, using some of our background sensing techniques as examples. It contributes a thorough way to think about the design space of failure modes, as well as the tradeoffs between different types of errors, that complements other work on cognitive "action slips" [Norman 1981].

What is important to recognize about this matrix is that it is impossible to completely avoid all four classes of error in a system design. A system that recognizes when the user holds the device like a phone has to balance the possibility of Type I errors (incidental handling triggering accidental recognition) versus Type II errors (failing to recognize the gesture within typical variations in the user's performance). But even if this balance is perfect, it can still suffer from Type III errors (if the user listens to the device for some other reason, this may be recognized as a desire to record a voice memo). The table also reveals that removing the sensors altogether cannot avoid errors: a system with no sensors can exhibit type I errors (pressing the record button by mistake during incidental handling of the device causes it to start recording) as well as type IV errors (it forces the user to explicitly control something that potentially

could have been automated, but is not; all of the burden to control the system is pushed back on the user).

These inescapable failure modes do not mean that good design is not possible. We can carefully consider each type of error and avoid any common and reasonable cases. Furthermore, we can implement mechanisms to recover from or mitigate failures when they inevitably occur, and we can give the user choices to customize or disable particular features. For example, the LookOut calendaring system [Horvitz 1999] gracefully degrades the precision of its automatically inferred services to match uncertainty about the user's goals by doing less, but doing it correctly. Bellotti et al. [2002] also expose challenges for dealing with "Accidents," raising issues such as how to control or cancel system action in progress; how to disambiguate what to undo in time; and how to intervene when the user makes an obvious error.

4.2 Mappings and Customization

Our system design uses a fixed mapping of the activities and gestures that it recognizes, to a set of services that it provides on the user's behalf. We chose services that seemed to provide the user with some benefit by being automatically sensed. However, allowing the user to configure or customize the system's response to various sensed states would be a useful facility, and will probably be necessary if these techniques find their way into widespread use. For example, users might prefer that their device play music at a private volume level [Hinckley and Horvitz 2001] or replay missed information [Dietz and Yerazunis 2001] when they "listen to it like a phone" rather than recording a voice memo; or users may want their device to automatically turn on and bring up their to-do list whenever they pick it up. Just as mobile devices allow users to reassign mechanical buttons, our system could allow customization of services provided via sensing techniques. To access customization features, users might manually access system settings (as is necessary to reprogram mechanical buttons), or alternatively it may be possible for the system to display a small icon, representing a sensed activity, for a short period whenever that activity is recognized. By tapping on this icon, the user could bring up a control panel that describes what is being sensed, and provides controls to customize the behavior. Such customization could include enabling or disabling individual features, adjusting timeouts, or setting, training, or calibrating the system with new trigger angles. This might allow users to compensate for unusual situations or usage contexts that we did not anticipate in our designs.

4.3 Caveats and Limitations

Despite careful design and consideration, our proposed techniques might fail to operate as expected. The sensors themselves have limitations. The tilt sensor performs poorly when the device is held on edge (its sensitivity follows an arcsine response curve, so the data is noisy at 90 degrees). If the user is wearing gloves, our touch sensor will not detect the presence of his hand. The proximity sensor performs poorly in direct sunlight or when bright spotlights shine on it. However, these problems have technical solutions: two tilt sensors could be

used instead of one; more carefully designed capacitance sensors could detect the presence of the user's hand through the gloves [Baxter 1997]; or a light sensor could be combined with the proximity sensor to partially compensate for the influence of ambient light [Hinckley 2003a]. It should also be possible for the system to attenuate or ignore the sensors when they are reporting noisy or unusual data, preferring to do less, but do it well.

A more significant problem is that users may exhibit unusual or unexpected ways of operating the device that may break the model of usage anticipated by our software, leading to incorrect inferences or failures to operate correctly. For example, if the user wishes to read while lying on his side, the orientation of the device as sensed by the system is off by 90 degrees from the orientation of device as seen by the user, and the system will display the screen in Landscape mode even thought the user may want Portrait mode. Users may need to intervene to remedy this type of problem.

For some users and tasks, even if our sensors operated flawlessly, the services that they provide might have limited value or interfere with direct user control. For example, when the user is already holding the PDA's stylus and fully attending to the mobile device, it may be quicker to explicitly start recording a voice memo using an on-screen widget rather than holding the device like a phone. The sensors may be inappropriate in situations with high error costs, highly trained operators, or tasks that require rapid, unambiguous results. By the same token, our sensing techniques seem especially well suited to mobile professionals with limited patience for manually controlling their devices, or for users who might be distracted by on-going conversations with colleagues yet still wish to have quick access to features of their PDA. In general, sensing techniques seem to offer some benefit when one or more of the following conditions hold:

The technique can eliminate or reduce user frustration and barriers to interaction by making it easier for the user to get at functionality while engaged in an auxiliary task or by making it easier to interleave short-duration tasks.

The technique can help mitigate attentional demands by supporting interaction techniques that enable eyes-free use via sensed activity, or by optimization of interaction in a real-time manner that would not be possible if the user had to explicitly manage details of the device. By moving responsibility for some actions and decisions from the user to the system, the user is free to concentrate on other things.

The technique expands the vocabulary of interaction for a mobile device which otherwise might be limited due to its small size. For example, a primary benefit of TiltScroll is that it enables one-handed horizontal and vertical scrolling.

5. CONCLUSIONS AND FUTURE WORK

Our research demonstrates how the foreground/background model can be applied to elucidate properties of sensing systems that complement those suggested by the five questions of Bellotti et al. [2002]. We have shown how the

model can play a role in the design of sensing systems by helping us reason about such systems, by classifying recognition errors in light of transitions in the ground, and even by forming the basis of architectural mechanisms that assist the implementation of sensing systems. We have identified sensing, reasoning about, and handling foreground/background states and transitions between the foreground and background as critical design issues for sensor-augmented mobile devices. Our experiences with our system have led us to propose nine lessons learned for foreground/background sensing systems:

L1. Use background sensing to assist users in transitions to the foreground.

L2. Preserve the user's focus of attention by minimizing disruptions to the ground.

L3. Provide feedback of transitions between the grounds and awareness of whether user activity will be interpreted as foreground or background.

L4. Scope foreground interpretation of possible interactions via background sensing.

L5. Automate blocking steps in the background by sensing and removing unnecessary barriers to interaction.

L6. Include ground selection mechanisms that help indicate whether activity should be interpreted as foreground or background.

L7. Prefer background interpretation for typical events.

L8. Provide explicit ground selection mechanisms that allow foreground techniques to dominate background interpretations.

L9. Explicitly encode ground transitions between foreground and background and use such encodings to minimize disruptive changes to the current mode of interaction.

Even if a system embodies all of these principles, we recognize that sensing techniques cannot offer a panacea for interaction with mobile devices, and careful design and tasteful selection of features will always be necessary. Only some of the actions that mobile devices support seem to lend themselves to solution via sensing techniques; other tasks may be too complex or too ambiguous, requiring some degree of human guidance or mixed-initiative problem solving.

A hybrid design integrating sensors with traditional techniques may prove to be the most practical approach: recalling the example of point-and-shoot photography, more advanced cameras provide dials, knobs, and other conventional means that allow knowledgeable photographers to choose which subsets of sensors to use and which to ignore or override. We currently do not provide such controls for our sensors, but we realize that they may become necessary as our techniques move from the research lab to real-world usage. Using sensors to deliver devices that are efficient and minimally distracting should not come at the expense of providing direct control when necessary.

A method of carefully quantifying the effects of various failure modes would be invaluable for sensing-based interaction, allowing systems to carefully balance the benefits of services against potential costs of recognition errors. It would also allow researchers to determine the value of information provided by a sensor, or to carefully measure the impact of different policies for interrupting the user. We are currently exploring the use of background/foreground

interaction in conjunction with decision-theoretic approaches to these problems [Horvitz 1999; Horvitz et al. 1999].

While interactive sensing techniques seem to provide many benefits, they also increase opportunities for poor design because the strengths and weaknesses in the design space are not as well understood as traditional interface design. We have contributed a number of examples of what is possible, and have suggested some general design principles, but future work needs to pursue careful experiments to quantify user performance with sensing techniques, as well as longitudinal studies to determine whether infrequent but persistent recognition problems may negate the apparent benefits of sensing techniques when they fail to operate as the designers intended.

Much previous research has focused on a few sensors that provide information for individual devices; but only a few sensing techniques, such as Pick and Drop [Rekimoto 1997], Smart-Its Friends [Holmquist et al. 2001], or Synchronous Gestures [Hinckley 2003b] have been designed for multiple user or multiple computer scenarios. Combining the capabilities of multiple independent sensing systems connected by wireless networking into a distributed sensing system may provide compelling new capabilities. Such distributed sensing systems may also make it practical to dramatically scale up the number and type of sensors that are available to individual systems. What becomes possible when a system can not only sense touch, tilt, and proximity, but can also detect its location, recognize real-world objects [Want et al. 1999], identify the user and other persons nearby, and see, feel, and hear through digital perceptual mechanisms? The emergence and integration of more comprehensive sensor systems may benefit not only individual users, but also networks of users and society as a whole.

ACKNOWLEDGMENTS

We are grateful to Eugene Shih for his efforts on our sensor hardware and firmware, as well as Andy Wilson and Steve Bathiche for discussions of sensors.

REFERENCES

BARTLETT, J. F. 2000. Rock 'n' scroll is here to stay. *IEEE Comput. Graph. Appl.* (May/June): 40–45.

BAXTER, L. K. 1997. *Capacitive Sensors: Design and Applications*. New York, The Institute of Electrical and Electronics Engineers.

BELLOTTI, V., BACK, M., EDWARDS, W. K., GRINTER, R., LOPES, C., and HENDERSON, A. 2002. Making sense of sensing systems: Five questions for designers and researchers. In *Proceedings of the ACM CHI 2002 Conference on Human Factors in Computing Systems*, Minneapolis, MN, 415–422.

BUXTON, W. 1995. Integrating the periphery and context: A new taxonomy of telematics. *Proceedings of Graphics Interface '95*, Quebec City, Quebec, Canada, 239–246.

DIETZ, P. and YERAZUNIS, W. 2001. Real-time audio buffering for telephone applications. In *Proceedings of the ACM UIST 2001 Symposium on User Interface Software & Technology*, Orlando, FL, 193–194.

DIX, A. 2002. Beyond intention: pushing boundaries with incidental interaction. In *Proceedings of Building Bridges: Interdisciplinary Context-Sensitive Computing*, Glasgow University, 1–6.

HARRISON, B., FISHKIN, K., GUJAR, A., MOCHON, C., and WANT, R. 1998. Squeeze Me, Hold Me, Tilt Me! An exploration of manipulative user interfaces. In *Proceedings of the ACM CHI'98 Conference on Human Factors in Computing Systems*, Los Angeles, CA, 17–24.

HINCKLEY, K. 2003a. Distributed and local sensing techniques for face-to-face collaboration. *ICMI-PUI'03 Fifth International Conference on Multimodal Interfaces*, Vancouver B.C., Canada, 81–84.

HINCKLEY, K. 2003b. Synchronous gestures for multiple users and computers. *UIST'03 Symposium on User Interface Software & Technology*, Vancouver, BC, Canada, 149–158.

HINCKLEY, K. and HORVITZ, E. 2001. Towards more sensitive mobile Phones. *ACM UIST 2001 Symposium on User Interface Software & Technology*, Orlando, FL, 191–192.

HINCKLEY, K., PIERCE, J., SINCLAIR, M., and HORVITZ, E. 2000. Sensing techniques for mobile interaction. *ACM UIST 2000 Symposium on User Interface Software & Technology*, San Diego, CA, 91–100.

HOLMQUIST, L., MATTERN, F., SCHIELE, B., ALAHUHTA, P., BEIGL, M., and GELLERSEN, H. 2001. Smart-its friends: A technique for users to easily establish connections between smart artefacts. *Ubicomp*, Atlanta, GA, Springer-Verlag, 116–122.

HORVITZ, E. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the ACM CHI'99 Conference on Human Factors in Computing Systems*, Pittsburgh, PA, 159–166.

HORVITZ, E., JACOBS, A., and HOVEL, D. 1999. Attention-sensitive alerting. In *Proceedings of UAI '99, Conference on Uncertainty and Artificial Intelligence*, Stockholm, Sweden, 305–313.

ISHII, H. and ULLMER, B. 1997. Tangible bits: Towards seamless interfaces between people, bits, and atoms. *Proceedings of CHI'97: ACM Conference on Human Factors in Computing Systems*, Atlanta, Georgia, ACM, New York, 234–241.

NIELSEN, J. 1993. Noncommand user interfaces. *Comm. ACM 36* (4): 83–89.

NORMAN, D. A. 1981. Categorization of action slips. *Psyc. Rev. 88* (1): 1–15.

PARTRIDGE, K., CHATTERJEE, S., SAZAWAL, V., BORRIELLO, G., and WANT, R. 2002. TiltType: Accelerometer-supported text entry for very small devices. *ACM UIST 2002 Symposium on User Interface Software and Technology*, Paris, France.

REKIMOTO, J. 1996. Tilting operations for small screen interfaces. *ACM UIST'96 Symposium on User Interface Sofware & Technology*, Seattle, WA, 167–168.

REKIMOTO, J. 1997. Pick-and-drop: A direct manipulation technique for multiple computer environments. *Proceedings of the ACM UIST'97 Symposium on User Interface Software & Technology*, Banff, Alberta, Canada, 31–39.

SAFFO, P. 1997. Sensors: The next wave of infotech innovation. *Institute for the Future: 1997 Ten-Year Forecast*, 115–122.

SCHILIT, B. N., ADAMS, N. I., and WANT, R. 1994. Context-aware computing applications. *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, IEEE Computer Society, 85–90.

SCHMIDT, A. 2000. Implicit human-computer interaction through context. *Personal Technologies 4* (2&3): 191–199.

SCHMIDT, A., BEIGL, M., and GELLERSEN, H.-W. 1999. There is more to context than location. *Comput. Graph. 23* (6): 893–901.

SELLEN, A., KURTENBACH, G., and BUXTON, W. 1992. The Prevention of mode errors through sensory feedback. *Hum. Comput. Inter. 7* (2): 141–164.

SMALL, D. and ISHII, H. 1997. Design of spatially aware graspable displays. *CHI'97 Conference Companion*, Altanta, GA, 367–368.

WANT, R., FISHKIN, K. P., GUJAR, A., and HARRISON, B. L. 1999. Bridging physical and virtual worlds with electronic tags. *Proceedings of the ACM CHI'99 Conference on Human Factors in Computing Systems*, Pittsburgh, PA, 370–377.

WIGDOR, D. and BALAKRISHNAN, R. 2003. TiltText:Using tilt for text input to mobile phones. *ACM UIST'03 Symposium on User Interface Software & Technology*, Vancouver, BC, Canada, 81–90.

Received February 2003; revised August 2003; accepted February 2004 by Shumin Zhai and Victoria Bellotti