Sensing Techniques for Mobile Interaction

Ken Hinckley, Jeff Pierce, Mike Sinclair, Eric Horvitz
Microsoft Research, One Microsoft Way, Redmond, WA 98052
{kenh, sinclair, horvitz}@microsoft.com; jpierce@cs.cmu.edu

ABSTRACT
We describe sensing techniques motivated by unique aspects of human-computer interaction with handheld devices in mobile settings. Special features of mobile interaction include changing orientation and position, changing venues, the use of computing as auxiliary to ongoing, real-world activities like talking to a colleague, and the general intimacy of use for such devices. We introduce and integrate a set of sensors into a handheld device, and demonstrate several new functionalities engendered by the sensors, such as recording memos when the device is held like a cell phone, switching between portrait and landscape display modes by holding the device in the desired orientation, automatically powering up the device when the user picks it up the device to start using it, and scrolling the display using tilt. We present an informal experiment, initial usability testing results, and user reactions to these techniques.

Keywords
Input devices, interaction techniques, sensing, context-awareness, mobile devices, mobile interaction, sensors

INTRODUCTION
The rapidly growing market for mobile devices such as personal information managers (PIM's: tablet, pocket, and credit-card sized), cellular telephones, pagers, watches, and wearable computers offers a tremendous opportunity to introduce interface design innovations to the marketplace. Compared to desktop computers, the use of PIM's is more intimate because users often carry or even wear PIM's throughout their daily routine, so they present HCI design opportunities for a more intimate user experience.

People also use mobile devices in many different and changing environments, so designers don't have the luxury of forcing the user to "assume the position"[1] to work with a device, as is the case with desktop computers. For example, the user must accept qualities of the environment such as light levels, sounds and conversations, and the proximity of people or other objects, all of which taken together comprise attributes of the *context* of interaction. But if mobile devices remain unaware of important aspects of the user's context, then the devices cannot adapt the interaction to suit the current task or situation. Thus an inability to detect these important events and properties of the physical world can be viewed as missed opportunities, rather than the basis for leveraging deeper shared understanding between human and computer. Indeed, Buxton has observed that much technological complexity results from forcing the user to explicitly maintain the context of interaction [3].

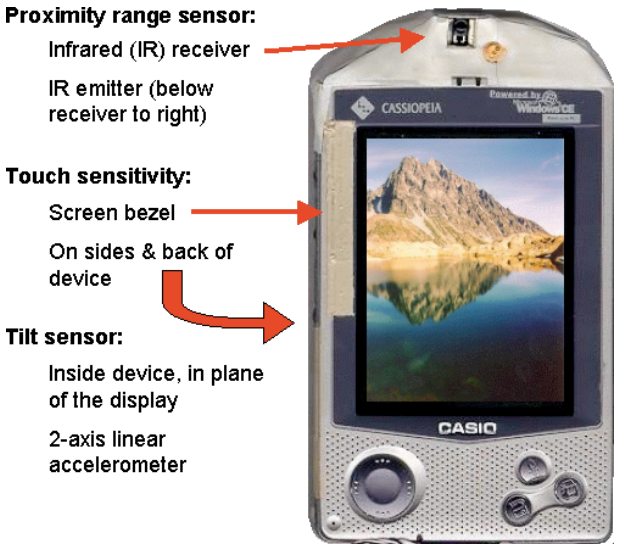


Fig. 1 Our prototype device, a Cassiopeia E105 Palm-sized PC. It is augmented with a proximity range sensor, touch sensitivity, and a two-axis tilt sensor.

Furthermore, the set of natural and effective gestures—the tokens that form the building blocks of the interaction design—may be very different for mobile devices than for desktop computers. Over the course of a day, users may pick up, put down, look at, walk around with, and put away (pocket/case) their mobile device many times; these are naturally occurring "gestures" that can and perhaps should become an integral part of interaction with the device. Because the user may be simultaneously engaged in real-world activities like walking along a busy street, talking to a colleague, or driving a car, and because typical sessions with the device may last seconds or minutes rather than hours [21], interactions also need to be minimally disruptive and minimally demanding of cognitive and visual attention.

We believe that augmenting mobile devices with sensors has the potential to address some of these issues. There is

ACM UIST 2000
Symposium on User Interface Software and Technology, CHI Letters 2 (2), pp. 91-100

Best Paper Award

[1] George Fitzmaurice made this observation and coined this phrase (personal communication).

an explosion of inexpensive but very capable sensors [9][18]. While these sensors may enable new interaction modalities and new types of devices that can sense and adapt to the user's environment, they raise many unresolved research issues. What interaction techniques or services can benefit from this approach? What problems can arise? What are the implications for end-users?

To explore some of these research issues, and work towards our design goal of providing context-sensitive interfaces that are responsive to the user and the environment, we have constructed a prototype sensor-enriched mobile device based on the Cassiopeia E-105 Palm-sized PC (fig. 1). We add a two-axis linear accelerometer (tilt sensor), capacitive touch sensors, and an infrared proximity range sensor. These sensors combine low power consumption and cost with the potential to capture natural, informative gestures.

We have sought to explore a range of interactive sensing techniques to gain experience with general issues and to explore issues of integrating techniques that may conflict with one another. We implement techniques such as voice memo recording by speaking into the device just as one would speak into a cell phone, switching between portrait and landscape display modes by holding the device in the desired orientation, automatically powering up when the user picks up the device, and scrolling the display using tilt. We suggest new points in the design space, contribute design and implementation issues and alternatives, and discuss challenges such false positive and false negative recognition. We present initial usability testing results and user reactions to these techniques, as well as an informal experiment that suggests our sensed gesture for voice memo recording may be less demanding of visual attention than traditional techniques.

RELATED WORK

Research in ubiquitous computing [27] has led to increased interest in providing system support for *background* interaction using passively sensed gestures and activity, as opposed to the *foreground* interaction of traditional GUI's. Buxton describes this vision and contributes a general foreground / background model of interaction [3].

An important part of enabling background interaction is to develop the sensors and software that can detect and infer information about the user's physical activity. For example, Harrison et al. [10] use pressure sensors to detect in which hand the user is holding a mobile device. Hinckley et al. [11] describe a touch-sensitive mouse. Zhai et al. integrate eye tracking with traditional manual pointing [30].

Sensors can also be used to augment or sense the environment itself. Want et al. [26] add electronic tags to objects and assign them unique ID's; a mobile device with a tag-reading sensor can then determine the identity of nearby objects. Rekimoto's Pick-and-Drop technique uses the unique identifier of each user's stylus to transfer information between devices [17].

Context awareness has been the subject of much recent research [5, 15, 19, 20, 22], with some ideas already appearing in commercial products (e.g., a light sensor for adjusting display quality [4]). Schmidt et. al. [22] describe a cell phone that combines tilt, light, heat, and other sensors to sense contexts such as sitting on a table, in a briefcase, or being used outdoors. These states modify the behavior of the device, such as the tone and volume of the ring. Schmidt et. al. have explored a number of other sensing techniques, including powering on/off a device based on touch, portrait vs. landscape display mode selection, and detection of walking [21][22][23], but they do not report usability testing, and many aspects of the interactive behavior still need to be further explored.

Horvitz et al. [13][14] describe architectures and techniques to infer attention and location via integration of sensed events (keyboard, mouse, and microphone). Sawhney & Schmandt [19] explore contextual notification. Schilit et al. [20] describe *proximate selection*, which uses location-awareness to emphasize nearby objects, making them easier for the user to select. Note that all of these techniques use background sensing to support foreground activity.

A number of research efforts have explored the use of sensors to provide additional input degrees-of-freedom for navigation tasks on mobile devices. Rekimoto uses tilting for menu selection and map browsing [16]. Harrison et. al [10], Small & Ishii [24], and Bartlett [1] use tilt sensors to scroll through and select information on a handheld device. The SmartQuill digital pen [28] uses tilt sensors to digitize the pen's ink trail. Fitzmaurice augments a palmtop device with a six degree-of-freedom tracker to create a virtual window into a 3D information space [7][8]. Verplaetse [25] reviews motion-sensing technologies.

HARDWARE CONFIGURATION AND SENSORS

All of our sensors and electronics are integrated directly into the Cassiopeia E105, making the device totally mobile. Digital and analog-to-digital inputs of a Microchip 16C73A Peripheral Interface Controller (PIC) microprocessor capture the sensor values. The PIC transmits the data to the serial port of the Cassiopeia. Also, our PIC processor remains powered up even when the Cassiopeia device itself is powered off. The software for our automatic-on feature executes in the PIC processor for this reason; all other features are implemented as Windows CE applications on the E105's processor. The PIC continuously samples the sensors and transmits packets to the host at 19200 baud (approximately 400 samples per second).

Touch Sensors

A large touch sensor covers the back surface and sides of the device, allowing us to detect if the user is holding the device. The sensor detects capacitance of the user's hand in a manner similar to [11], except the sensor is divided into two regions (an "active" area and a "ground" area) because we encountered problems detecting capacitance to a single sensor pad on a small mobile device. We placed a second touch sensor on the left side of the screen bezel.

Tilt Sensor

Our device currently uses an Analog Devices ADXL05 two-axis linear accelerometer. This sensor detects the tilt of our device relative to the constant acceleration of gravity. This sensor also responds to linear accelerations, such as those resulting from shaking the device. Figure 2 shows some example data of one of the authors entering an elevator, looking at the display, holding the device down at his side, and finally walking to a meeting.

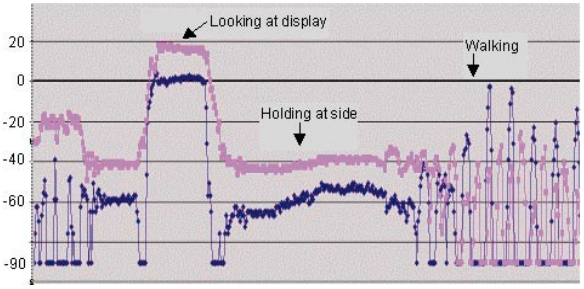


Fig. 2 Example tilt data. The top trace is forward/back tilt; the bottom trace is left-right tilt.

The tilt sensors are most accurate when held flat, and become increasingly insensitive to tilting as the angle approaches 90°. They follow a response curve of the form $\text{Angle} = \sin^{-1}((T - T_c) / K)$, where T is the tilt sensor value, T_c is the sensor value at 0°, and K is a gain parameter. Because the sensor cannot detect the sign of the gravity vector, it is unable to determine if the user is holding the device with the display facing right side up, or upside-down. We could augment the sensor with a simple gravity-activated switch to work around this limitation, but we have not yet implemented this. One other limitation of the tilt sensor is that it cannot respond to rotation about the axis parallel to gravity. Adding a digital magnetic compass, as found in some mountaineering watches, may allow us to overcome this missing degree of freedom in future work.

Proximity Sensor

The proximity sensor uses an infrared transmitter / receiver pair positioned at the top of the device (fig. 1). A timer drives the transmitter, an IR light-emitting diode with a 60° beam angle, at 40 kHz. The IR receiver is same type typically used to receive remote control signals. These receivers have an automatic gain control output that we use to measure the strength of the received signal. With our emitter/detector pair placed close together on the device, the receiver senses the reflected IR light off of the user's hand or other object; this signal is proportional to the distance to the object. Fig. 3 shows the sensor response.

We calibrated this sensor by measuring the actual distance to an author's hand in a normally lit office environment. As seen in the graph, the sensor response reaches a maximum at approximately 5-7cm from the sensor, and does not increase further if the user or an object moves closer; even if the user is actually touching the sensor it still returns the maximum value. Beyond about 25cm the data is noisy. Dark objects reflect less light and appear further away; ambient light can also affect the readings, although in practice we have found that only direct sunlight is truly problematic, reducing the range to only a couple of inches.

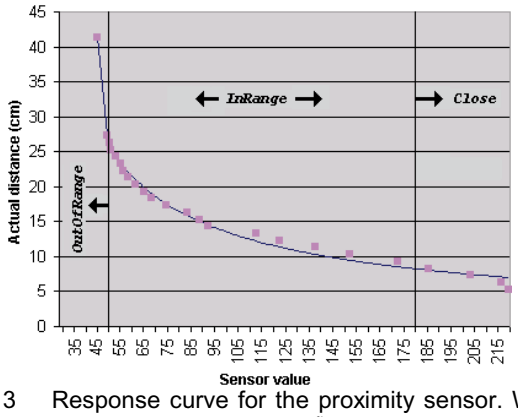


Fig. 3 Response curve for the proximity sensor. We use the curve $Z_{cm} = K/((P/P_{max}) - c)^{\alpha}$ to approximate the data. Z_{cm} is the distance in cm, P is the raw proximity reading, P_{max} is the maximum sensor reading, c is a constant, α is the nonlinear parameter (0.77), and K is a gain factor.

Our proximity sensor currently consumes more power than we would like it to, but we could reduce power consumption by only pulsing the LED a few times a second when the user is out of proximity, or by reducing the duty cycle of the 40kHz IR LED output.

SOFTWARE ARCHITECTURE

We implemented a software context information server that acts as a broker between the PIC / sensors and the applications. The server continuously receives sensor data packets from the PIC, converts the raw data into logical form, and derives additional information (fig. 4). Applications can access the context data by polling a block of shared memory where the context server maintains the latest context information, or alternatively, by asking the server for notification when a specific piece of information changes value. We implement this functionality by sending messages between applications. We also allow applications to share information by submitting it to the context server.

We use the names of the context variables shown in fig. 4 to help describe our interaction techniques. Names in the `Courier` font represent context variables (which can also be thought of as events). *`Italicized`* items represent particular named values of a context variable.

INTERACTIVE SENSING TECHNIQUES

Creating smarter interfaces by giving computers sensory apparatus to perceive the world is not a new idea, but nonetheless there are few examples of interactive sensing techniques. By implementing specific examples, we explore some new points in the design space, uncover many design and implementation issues, and reveal some preliminary user reactions as well as specific usability problems.

Usability Testing

In the following sections, we discuss usability issues in the context of each technique. Seven right-handed test users (2 women, 5 men) between the ages of 30 and 50, all current

users of palm-sized PIM devices, participated in our informal usability tests. Four own Palm Pilots, and 3 own Windows CE devices (2 Casio E100 series, 1 Philips Nino). The occupation of most participants required significant mobility; some used their devices to store commonly needed files, while others claimed, "it controls my life."

	Context Variable	Description
Touch	`Holding & Duration`	Whether or not user is holding the device, and for how long. (direct reading of touch sensor)
	`TouchingBezel, Dur`	If the user is touching the screen bezel, and for how long. (bezel contact over 0.2 sec.)
Tilt/Accelerometer	`TiltAngleLR, TiltAngleFB`	The left/right and forward/back tilt angles, in degrees. (sensor reading & transform per fig. 3)
	`DisplayOrientation & Refresh`	*`Flat`*, *`Portrait`*, *`LandscapeLeft`*, *`LandscapeRight`*, or *`PortraitUpsideDown`*. A `Refresh` event is posted if apps need to update orientation.
	`HzLR, MagnitudeLR, HzFB, MagnitudeFB`	Dominant frequency and magnitude from FFT of tilt angles over the last few seconds.
	`LookingAt & Dur.`	If user is looking at the display.
	`Moving & Duration`	If device is moving in any way.
	`Shaking`	If the device is being shaken vigorously.
	`Walking & Duration`	If the user is walking.
Proximity	`Proximity`	Estimated distance in cm to proximal object, if in range. (sensor transform per fig. 4)
	`ProximityState & Duration`	*`Close`*, *`InRange`*, *`OutOfRange`* (see fig. 4), *`AmbientLight`* (when out-of-range and bright ambient light is present).
Other	`Scrolling`	If the user is currently scrolling. (posted by scroll app)
	`VoiceMemoGesture`	If recording a voice memo. (posted by voice recording app)

Fig. 4 Some of the sensor data & derived events that are available from the Context Server.

VOICE MEMO DETECTION

Some current PIM devices include voice recording features, and many dedicated digital voice recorders are available on the market. However, finding a 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 PIM like a cell phone or microphone and speaking into the device– a natural, implicit gesture that requires little cognitive overhead or direct visual attention. This gesture allows our PIM to have a very specific sensed context of use, resulting in a combination of a general-purpose device with many capabilities, and an appliance-like device with a specific use.

The user's impression is that one just speaks into the device to make it record. Our implementation of this concept uses all three of our hardware sensors:

- The user must be holding the device. This prevents accidental activation when in a purse or briefcase.
- The user must hold the device in Close proximity, or within approximately 8 cm, to speak into it.
- The user must tilt the device towards himself. This is the natural posture that the hand makes when bringing an object towards the head. Fig. 5 describes the exact criteria for acceptable angles.

If these conditions hold true for 0.1 seconds, the device makes a distinct click (to give early feedback that the gesture has been recognized), and starts the standard WinCE voice recorder control. The control issues a single sharp beep just before it starts recording, after which the user can leave a voice memo of any length. When finished speaking, users naturally move the device away, which automatically stops the recording. We stop recording if the device enters the proximity *`OutOfRange`* state, if it returns to a mostly flat orientation (±25°), or if the user stops `Holding` it. The voice recorder control issues two sharp beeps when recording stops. The audio feedback seems crucial to the interaction, as it provides non-visual feedback of the gesture recognition, cues the user when to start speaking, and confirms that the memo was recorded.

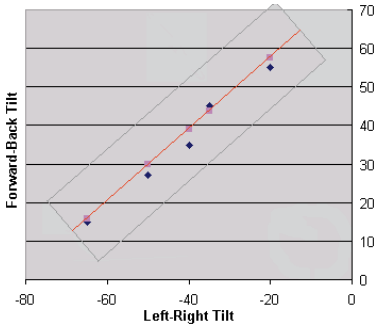


Fig. 5 Acceptable angles for voice memo detection (device in left hand). The candidate angle must fall within ±10° of the line *segment* shown above. We collected candidate samples by using the device in either hand. The same model, but with a negative slope, fits the right-handed poses. The model is y = mx + b with m=0.925 and b=76.

Informal Experiment

To explore our hypothesis that the sensed voice memo gesture requires less cognitive and visual attention than traditional methods, we collected some quantitative data by asking our test users to perform a visual tracking task. This tracking task was used to simulate a visually intensive real-world task, such as driving. The data are suggestive but not conclusive. We studied three separate conditions:

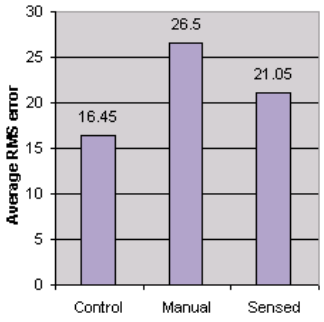
Control (C): For one full minute, the subject attempted to track a pseudo-randomly moving cross symbol, which was

displayed on a traditional computer monitor, using a standard computer mouse. We generated the motion using summed sinusoidal functions, as typically done in manual tracking experiments [29], with an amplitude of 100 pixels and a base frequency of 0.06Hz.

Sensed (S): The subject performed the tracking task with the mouse in the right hand, while simultaneously holding the E105 device in the left hand and recording voice memos ("Testing, 1-2-3") using our sensed gesture. We required the user to put the device down on a desk, and then re-acquire it, after recording each message. The user recorded as many messages as possible during a 1-minute trial, while simultaneously tracking the moving cross symbol.

Manual (M): As above, except the subject used the E105's built-in recording button to record the voice memo. The button (6mm in diameter) is located on the left side of the device, and it must be held down while recording.

All subjects performed the control condition first. We counterbalanced the Order of the Sensed and Manual conditions. One subject was not able to attend the study, so as a result we have 7 users (4 Manual first, 3 Sensed first). The user clicked at the center of the cross to start the trial. At 100Hz, we calculated the RMS (root mean square) error between the mouse position and the cross symbol, and then updated the position of the tracking symbol. We used the average RMS error (in pixels) over the course of the trial as the outcome measure. Fig. 6 shows the results.



Average RMS error		
	Avg.	std. dev.
C	16.45	0.95
M	26.5	7.19
S	21.05	2.24

RMS error by Order		
	M 1st	M 2nd
C	16.86	15.89
M	26.95	25.91
S	22.49	19.12

Fig. 6 Results of informal experiment. The tables show the average RMS error (in pixels) and standard deviation for each condition, as well as the RMS error by Order (whether the subject performed the Manual condition first or second).

The Manual condition exhibited the worst average performance, with 61% more RMS error than the Control condition, and 25% more error than the Sensed condition. The Sensed condition exhibited 27% worse performance than the Control condition. Two-tailed t tests revealed that both the Manual condition ($p<0.01$) and the Sensed condition ($p<0.001$) differed significantly from the Control condition. However, although the averages are suggestive, and six out of the seven subjects reported that the Sensed condition requires less concentration, the statistical difference between the Manual and Sensed conditions was marginal ($p=0.097$, not significant). This results from the small number of subjects and the high variance in the Manual condition, which we believe occurred due to differing subject strategies and pace recording voice memos. For a more definitive result, we would need to devise a method of more carefully controlling the pace and level of performance for the actual voice memo recording. Nonetheless, although one test subject did prefer the Manual button, the current data is quite suggestive that the sensed technique may require less cognitive or visual attention. Future studies will need to resolve this issue.

Usability Problems & Other Observations

The main usability problem with the sensed gesture is that it is not easily discoverable. Current users do not expect devices to be able to react in this way. However, the only instruction subjects needed to use it was "talk into it like you would talk into a cell phone."

Several test users commented that the sensed gesture was "Quite a bit easier, I can focus on what I'm trying to do" and that they "would probably use the voice recorder more if it worked that way." Users did not think that the gesture was necessarily any faster, 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." Figure 7 shows our analysis of the workflow for voice recording; the sensed gesture seems to better support the user goal **Record a message** by naturally phrasing the task into a single cognitive chunk [2].

Normal Button Hold	Sensor-Based Gesture
1. Pick up device 2. Find the ¼" button 3. Position hand to press button 4. Press & maintain tension 5. Listen for beep **6. Record message** 7. Release button 8. Double-beep confirms	1. Pick up device (to face) 2. Listen for click, beep **3. Record message** 4. Relax device when done 5. Double-beep confirms completion

Fig. 7 Workflow analysis of the voice recording interfaces. Subjects particularly felt that concentration was required to find and acquire the button, and then remember to maintain continuous tension on the button (steps 2, 3, and 4).

Overall, 6 out of 7 participants preferred the sensed gesture to using the button (average 4.29 on 5-point Likert scale). One user did not like the sensed gesture at all, commenting that it was "disorienting to put up to my face to talk." We did observe two instances where false positives occurred: one user triggered voice recording when demonstrating how she might put the device in a sweater pocket; another held the device with her hand on top of the display while walking, triggering recording when she tilted it at an angle. This latter false-positive condition could be eliminated if we looked for a *transition* in the proximity from `InRange` to the `Close` state (this currently is not required); the previous case seems harder to eliminate, although it should be noted that the memo turned off as soon as she dropped the device in her pocket (since `Holding` is required). Also, keep in mind that the traditional button solution itself suffers from false positive (hitting it by mistake) and false negative (forgetting to hold down the button) conditions.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.