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].

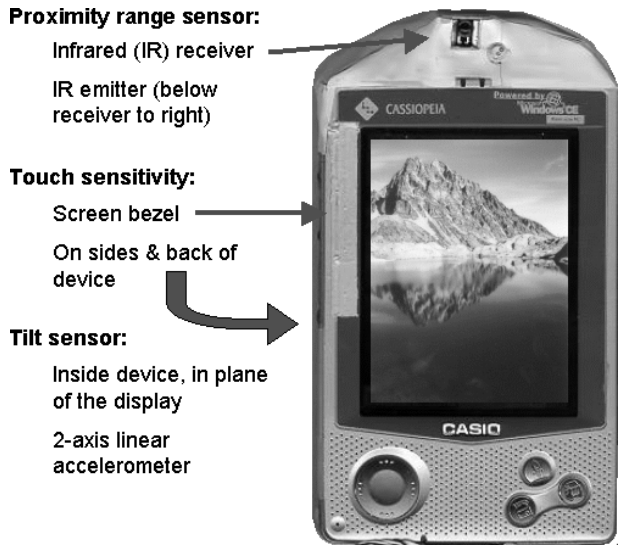


Fig. 1 Our prototype device, a Cassiopeia E105 Palm-sized PC. It is augmented with a proximity range sensor, touch sensitivity, and a two-axis tilt sensor.

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
UIST '00. San Diego, CA USA
© 2000 ACM 1-58113-212-3/00/11... $5.00

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

Petitioner Samsung Ex-1054, 0001

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.

Petitioner Samsung Ex-1054, 0002

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.

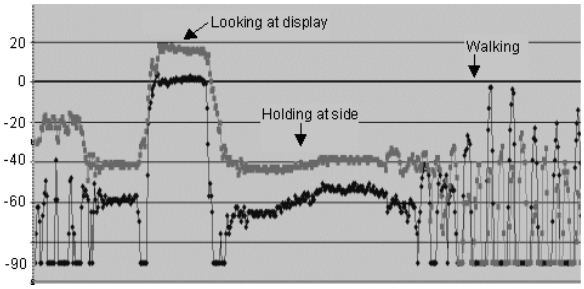


Fig. 2 Example tilt data. The top trace is forward/back tilt; the bottom trace is left-right tilt.

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.

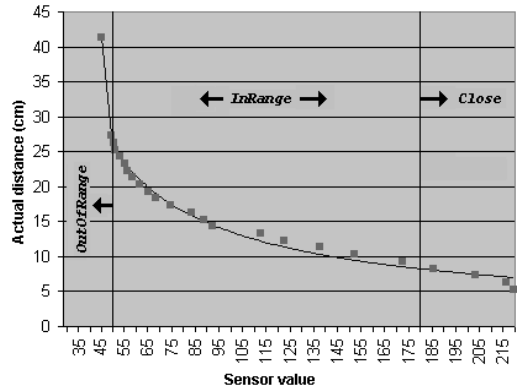


Fig. 3 Response curve for the proximity sensor. We use the curve $Z_{cm} = K/((P/P_{max}) - c)^{\alpha}$ to approximate the data. Z_{cm} is the distance in cm, P is the raw proximity reading, P_{max} is the maximum sensor reading, c is a constant, α is the nonlinear parameter (0.77), and K is a gain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

Petitioner Samsung Ex-1054, 0003

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 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.

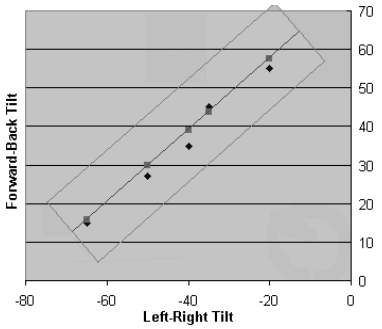


Fig. 5 Acceptable angles for voice memo detection (device in left hand). The candidate angle must fall within ±10° of the line *segment* shown above. We collected candidate samples by using the device in either hand. The same model, but with a negative slope, fits the right-handed poses. The model is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

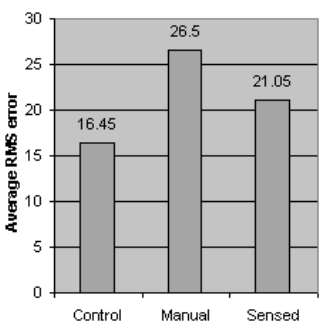
Petitioner Samsung Ex-1054, 0004

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.

Petitioner Samsung Ex-1054, 0005

PORTRAIT / LANDSCAPE DISPLAY MODE DETECTION

Unlike a stationary desktop monitor, users of mobile devices can tilt or rotate their displays to look at them from any orientation. Using the tilt sensor, we detect these gestures and automatically reformat the display to suit the current viewing orientation. For example, a user reading an E-book or inspecting a spreadsheet may find a portrait or landscape display mode more pleasant depending on the document content.

When the user holds a palm-sized device, he will naturally tilt it towards himself. We process these tilt angles and format the window to the nearest 90 degree rotation. Note that, assuming a non-rectangular display, simply rotating the bitmap is not always sufficient, as seen in fig. 8; the user interface must reformat itself to accommodate the display orientation, as suggested by Fitzmaurice et al's work with Rotating User Interfaces [6]. Our application also rotates the orientation of some other inputs (in this case, the direction pad, which provides previous / next page navigation) to maintain correspondence with the display.

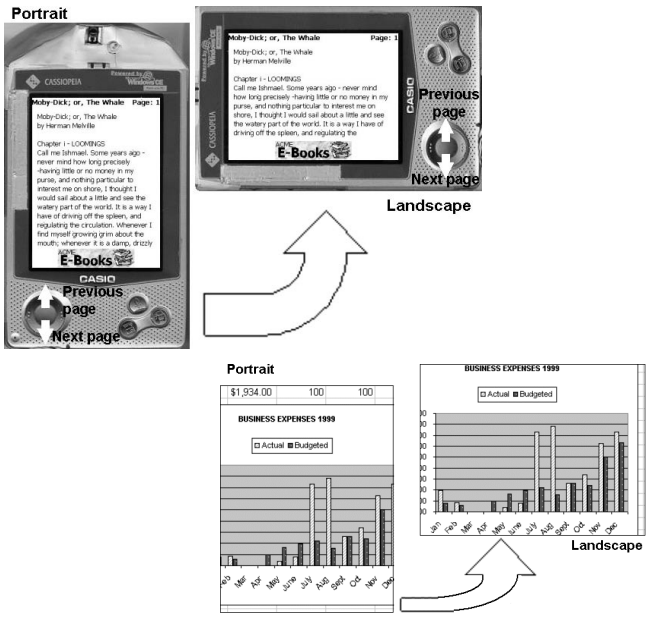


Fig. 8 Portrait / Landscape display mode detection. **Top**: An E-book application. The display automatically rotates and reformats the UI to fit the new screen orientation. **Bottom**: Spreadsheet application. The user can get the most out of the small display.

As other examples, a digital camera could sense the orientation at which a photograph was captured (we did not implement this), or a drawing program could reformat its screen real estate to accommodate the desired proportions of a sketch. We did implement such a drawing program, but it was not presented to our test users. However, if one wishes to rotate the display to allow drawing a curve from a comfortable angle (as experienced artists do constantly [6]), the users must place the device flat on a desk surface, lest the display reformat itself at an undesired time.

Fig. 9 shows how we convert the sensed tilt angles to a display orientation. The gray zones are ±5° dead bands that prevent jitter; to change display orientation, the tilt angles must pass all the way through a gray region until they fall into one of the four outside white regions, and the angles must stay within the same white region for 0.5 seconds. When both tilt angles fall within the center region (±3°), we consider the device to be resting `Flat` and do not change the display orientation.

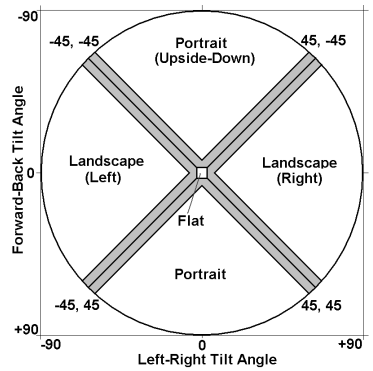


Fig. 9 Plot of left-right tilt vs. forward-back tilt and the sensed orientation.

An important implementation detail is to use the center of the screen (C_x, C_y) as the center of rotation when rendering the screen. Otherwise, the rotation when the user switches display modes may not appear intuitive. The transform of a point (x, y) in the document to a point (x', y') on the screen is given by $M=T*R*T^{-1}$, where T is the translation ($-C_x$, $-C_y$) and R is the 2D rotation matrix (for 0°, 90°, 180°, or 270°).

The "put-down problem" arises when the user places the device on a flat surface: the user may tilt the display while setting it down, which can change the display mode unintentionally. One solution is to simply extend the time-out to switch display modes from our default 0.5 seconds to 2 or 3 seconds, but this introduces annoying lag into the process when a display mode switch *is* desired. Instead, we maintain a FIFO queue of recent display orientations. When the user puts down the device (indicated by `Flat` and not `Holding`), we search through this queue to find the most stable recent orientation (other than `Flat`). A `Refresh` event is sent out if the display mode needs to be changed. The net result is that the device has a strong tendency to maintain its current display mode when the user puts it down. We had test users try picking up and putting down the device several times, and users clearly felt that this was the expected behavior: they did not expect it to revert to `Portrait` mode, for example.

All users felt that it was easy to switch display modes by turning the display (average rating of 5). One user described the technique as being "like a snow globe" and explained that it was "so easy to change direction I would probably use the other [display] modes, like to show the screen to someone else." For comparison, we also had users try switching the display using a traditional menu that dropped down from the menu bar. When asked if "I prefer

Petitioner Samsung Ex-1054, 0006

to switch the display mode using the drop-down menu" the average rating was 2 (disagree). Six of the seven users preferred the sensed gesture, while one user disliked the technique: "I think it would drive me nuts… I liked it better when I had control of it."

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 (thus switching to the `PortraitUpsideDown` mode). 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.

Schmidt proposes an automatic Portrait / Landscape technique where "the screen orientation is adapted to device orientation whenever a stable change in orientation is sensed," but provides no other description of the behavior or user-level issues. Bartlett [1] switches display modes if the user stands the device on edge for about 2 seconds. We use a different algorithm to quickly determine the orientation and contribute another approach to integration with tilting for scrolling the display as described below.

TILT SCROLLING & PORTRAIT / LANDSCAPE MODES

The idea of tilting to scroll the display of a mobile device has been well documented by previous work [1][10][16]. We contribute several issues to consider in the signal handling, as well as some novel twists. Due to time constraints, only 5 of our 7 users tried tilt-scrolling.

Clutching and Screen Real Estate Optimization

We use contact with the screen bezel (`BezelTouching`) to initiate scrolling. Scrolling continues until the user releases contact. An advantage of using this touch sensor to engage scrolling is that the sensor has a large surface area and does not require muscle tension to maintain contact. However, inadvertent contact can be a problem, so a large, flat traditional button or pressure sensor [10] may be preferable. Bartlett [1] uses a tilt gesture that locks the display, allowing scrolling without a clutch.

Several previous systems set a predefined or user-selectable "zero orientation" relative to which the scrolling takes place. Our system instead uses the orientation when the user initiates scrolling, allowing use of the device in any display mode and almost any comfortable posture.

We also observed that the application menu (at the top of the screen) and the Start bar (at the bottom of the screen) are not useful while scrolling. Therefore, while the user is scrolling, we hide these widgets. They reappear when the user releases the bezel. The user can also touch the bezel, without tilting, to view a document in "full screen mode."

Transfer Function

We have experimented with several transfer functions for calculating the rate of scrolling from the tilt angles, all of the basic form $v_{fb} = K * \mathbf{sgn}(dA_{fb}) * \mathbf{max}(||dA_{fb}|| - dA_{min}, 0)^{\alpha}$ where v_{fb} is the calculated velocity for the forward-back scrolling axis, K is the control gain, dA_{fb} is the change in angle from the zero orientation, dA_{min} is the size of the dead band, and α is the nonlinear parameter. We currently use $K=0.2$ and $\alpha=1.6$, but we have not yet performed an optimization study. We calculate the v_{lr} velocity for left-right tilting by substituting the left-right tilt values in the above equation. We set the dead band dA_{min} to 4° to allow the user to hold the bezel to see the full screen mode, without causing any scrolling; without this feature, a dead band of 2-3° would be probably be preferable.

We implemented several variations of this basic transfer function, each of which has its own advantages:

Mixed Axis Control: In the above equation, the dead band cutoff actually has a square shape when considered in two dimensions, giving this input mapping a slight affinity to move along the principle axes when moving slowly, which aids precision adjustment.

Dual Axis Control: We adjust the dead band size by normalizing (dA_{fb}, dA_{lr}) to a unit vector and multiplying the resulting components by dA_{min} for each axis. This results in a circular dead band region and thus more fluid motion when moving across both tilting axes.

Single Axis Control: This method allows motion to occur along only one principle axis at a time. We apply the above equation only to the axis corresponding to the maximum of ($||dA_{fb}||$, $||dA_{lr}||$). The velocity component of the other axis is set to 0. This allows even rapid scrolling across a long distance to be accomplished without drifting off-axis, at the cost of making diagonal movements more difficult. However, we feel this approach eases the required dexterity for tilt-scrolling, and helps prevent the user from becoming "lost" in a large 2D space because of unintended drift. This behavior also best fits the spreadsheet demonstration employed in our usability study, so all users tried tilt-scrolling with this transfer function (time did not permit user trials of the other transfer functions). Bartlett [1] appears to use a similar method to our single-axis control, based on his videotape (on the web).

We felt some concern that trying to use tilt-scrolling while walking with the device might cause problems; as seen in fig. 2, linear accelerations can significantly affect the tilt angles while walking. Fortunately, if the users walks while *looking at* the display, the linear accelerations are nearly perpendicular to the display and therefore the contamination of the tilting signal is minor, and only presents a problem if the user wants to make precise scrolling motions (on the order of < 5-10 pixels).

Contrast Compensation

Like most LCD screens, the apparent contrast of the TFT (Thin Film Transistor) active matrix LCD of the E105 changes with the viewing angle. As a result, the screen may appear dim or washed-out as the user tilts the display to

Petitioner Samsung Ex-1054, 0007

scroll. We generated an approximate calibration curve for this contrast effect (fig. 10) by viewing the screen from different angles and manually adjusting the contrast. The E105 provides 10 levels of contrast (0-9). We approximate the results by the equation $\text{Contrast} = m * dA_{fb} + b$ with $m = -0.135$ and $b = 5.25$.

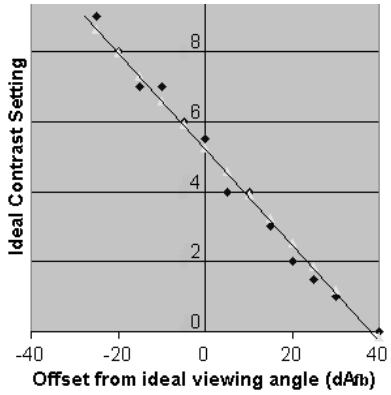


Fig. 10 Forward-back viewing angle vs. contrast setting.

We assume the user starts scrolling from a good viewing angle, and then apply the above model to adjust the contrast while the user is scrolling. To prevent hysteresis, we require the tilt angle to change more than 3° from the previous contrast setting before readjusting it. When the user stops scrolling, we continue to compensate for the contrast until the tilt of the display returns to the zero orientation of the original scrolling action (hence returning contrast to its original value). Resetting the contrast immediately when the user stops scrolling can cause an undesired drastic change. Even with our current approach a subtle flashing of the screen is occasionally visible; we could lessen this effect if more intermediate contrast settings were available on the hardware (although none of our test users noticed anything). We currently compensate the contrast along only the forward-back tilt axis, as left-right tilting has little effect on the apparent contrast.

Integration of Portrait / Landscape Modes

Integrating of tilt-scrolling and portrait / landscape mode detection presents some difficulties because the interaction techniques use tilting to control two different actions. Obviously, the screen should not change orientations during tilt-scrolling; our scrolling application shares the `Scrolling` event with the context server so that orientation-sensitive applications can determine if the user is currently scrolling. We also rotate the scrolling to be consistent with the current `DisplayOrientation`.

When the user stops scrolling (by releasing the screen bezel), the screen may be tilted towards a new display orientation. The screen should not change display modes at this point, as it would be disconcerting. Our tilting application currently remembers the logical display mode when scrolling ends (as opposed to the display mode currently seen on the display, which has remained fixed during scrolling) and only updates the display mode when it receives an event for a new and *different* orientation. However, one problem remains with this policy: if the user *wants* to switch to that orientation after he stops scrolling, it is unnatural to move the display to some *other* orientation and then back to the desired one. All 5 users who tried the combination of scrolling and portrait/landscape detection encountered this usability problem. We believe that instead, the display mode should switch to the orientation of the device when scrolling completes, *if* the display is held there beyond a certain time out (perhaps 1 or 2 seconds), although this change needs to be tested.

Overall, when asked if "I felt that tilting was a good way for me to scroll the screen" the average response was 4.8. For comparison, we implemented scrolling on the Cassiopeia's built-in direction pad. When asked if "I would rather use the direction pad to scroll a large document" the average response was 1.8 (disagree). Most users felt that tilting was more intuitive than the direction pad, and several commented that it was easier to operate one-handed: "It's easy and it leaves my other hand free"; or, "it simplifies the movement, I'm not trying to find the right button." Several users who liked it overall had at least one negative comment, such as "nice but sometimes overshoots," "takes a bit to get used to… but didn't take long" and "It helped if I envisioned the spreadsheet with weight."

POWER MANAGEMENT

Finding the power button on mobile devices can often be physically awkward and demanding of attention. The power button placement, size, and protrusion must balance the desire to have it easily accessible against the danger of accidentally hitting the power button or having it held down while carrying the device in a pocket (our test users reported several anecdotes of draining their batteries).

We observed that people typically hold their PIM in a posture appropriate for use, and *then* press the power button. We eliminate the need for this explicit button press as follows: when the power is off, if the user is `Holding` the device and `LookingAt` the display with a `DisplayOrientation` of *`Portrait`* (but not *`Flat`*), and this state persists for 0.5 seconds, then the PIC powers up the device. This results in the following behavior:

- The device cannot power up when in the user's pocket or purse because the user is not holding it.
- 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 `LookingAt` the display, which we set to true whenever (1) it is not *`Flat`*, (2) the left-right tilt is between ±15°, and (3) the forward-back tilt is greater than -5°. At present, we have implemented automatic power-up only for the *`Portrait`* display orientation.
- The timeout prevents it from powering up due to transient signals, but is short enough that the user does not have to wait for it. Based on our usability testing, this timeout should be slightly shorter that it is now.

Petitioner Samsung Ex-1054, 0008

All users were able to discover this feature, because as soon as they picked up the device to try to use it, the device would turn on. We also asked test users to "walk with the device the way you normally would if you were carrying it with you, but *not* intending to use it." Two test users experienced a false-positive resulting from the tilt sensor's inability to recognize up from down: if the user holds the device with the back of the device facing upward, the device powers on. We believe this problem could be addressed by adding a gravity-activated switch to disambiguate the orientation. The problem arose when the subjects held the device upside-down against their lap as they got up from a chair, although one said he "didn't care" because he would put the device away if he wasn't using it.

Our device also uses the touch, proximity, and tilt sensors to prevent undesired power-off or screen dimming due to the default system inactivity time-outs. If the device is already on, while the user continues `Holding` the device, we assume that the user must have left the device on for a reason. Thus we prevent the device from powering off (via the WinCE *SystemIdleTimerReset* function). We also use `Proximity` as a secondary indication of user activity if the device is lying flat on a table. This is useful if the user is still referring to the display, but not currently "using" the device, such as when using its calculator to compute dimensions for a woodworking project. We reset the idle timer if proximal (*`Close`* or *`InRange`*) *motion* is sensed; a proximal, but unchanging, signal is ignored after 15 seconds, which prevents an object left on or near the device from maintaining the power indefinitely.

Overall reactions to these features were positive. One user commented, "I love it! I'm constantly hitting the power button," while another said that "I prefer it to the button but it's not a huge thing." When asked if "I would rather press the button than have the device turn on for me" the average rating was 2.14 (disagree). One user preferred the button.

Some existing devices provide functionality similar to our automatic power-up through other means. For example, the Nokia 6162 cell phone answers a call (and turns on the display backlight) when the keypad cover is opened. Beverly Harrison demonstrated a prototype Softbook at UIST'99 that turned on when the user opened the book cover. Our device provides similar functionality without requiring the user to open or close anything (allowing easy one-handed operation). Schmidt [21] describes an on/off technique where "The user has the device in her hand. In this case the application is switched on, if the user is putting the device out of her hand it is switched off after a certain time," but does not give further details.

DISCUSSION

While integrating individual interaction techniques into a coherent system is a hard problem in general, we believe this can be especially true when integrating interactive sensing techniques. Several times we found that we needed to adapt our interaction techniques to prevent undesired interactions (e.g. tilt-scrolling and tilt display mode selection). While we believe that our current prototype integrates the interaction techniques fairly well, we cannot guarantee that our prototype will continue to scale well as we add new interaction techniques.

Experience with our prototype leads us to believe that sensor fusion (aggregating the data from multiple sensors) will be essential to expand its capabilities and support additional interactive sensing techniques. For example, an early version of our voice memo sensor used only the device tilt angles to recognize the gesture. It suffered from many false positive conditions if we made the gesture casual and easy to perform, or many false negative conditions if we made it more difficult to perform by reducing the parameter space of the gesture. However, by using multiple sensors together, we can provide a gesture that is unique enough that false-positives are uncommon, but easy to perform so that false-negatives are also rare. Of course, adding more sensors may increase size, weight, cost, or power consumption, so future research will need to determine the optimum balance between these factors.

Resonating with other work on autonomous UI decisions [12], our work also demonstrates the need for a careful analysis of possible false positives and negatives in the interface design. We do not believe that sensing is necessarily annoying if wrong, but designers need to be careful to design for graceful failure in the event of incorrect inferences. A method of quantifying the effects of positive and negative failures would be invaluable for this effort, allowing researchers to determine if, for example, providing users with a mental model of how the sensors work helps mitigate failure. A quantitative approach to measure the results of failure would also help us determine when the effects of an incorrect inference are so bad that the system should not even try. We plan to explore decision-theoretic approaches to these problems [12][14].

CONCLUSIONS AND FUTURE WORK

Contextual awareness via sensing may be increasingly relied upon for future interfaces to mobile devices. Exactly what can and cannot be accomplished with such techniques is not yet well understood. Examples contributed here and by other researchers suggest that simple, cheap, and relatively dumb sensors may play an important role in the future of mobile interaction. Nonetheless, we must also recognize that sensing techniques cannot offer a panacea for UI on 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. A hybrid design integrating sensors with traditional techniques may prove to be the most practical approach, if only because some unexpected cases may defeat the sensors. Yet overall, we believe that careful use of sensors can help deliver devices that are as simple and pleasant to use as possible while still allowing direct control when necessary.

Petitioner Samsung Ex-1054, 0009

In future work we would like to support contextual interaction, display, and notification services, all of which can benefit from awareness of the user's activity [14][19]. For example, our prototype device can detect `Walking` by using a Fast Fourier Transform at 10 Hz with a window of 32 samples. Walking exhibits a dominant frequency between approximately 1.4 Hz and 3Hz; we look for a continuous signal of sufficient magnitude that lasts for at least 4 seconds. Some manipulative signals can look like walking, but shared events (`Scrolling` or `VoiceMemo-Gesture`) eliminate most false-positives. This algorithm correctly detected walking for all of our test users, suggesting that we could use it to help inform activity-sensitive notification services.

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 GUI design. We need experiments to quantify user performance with these techniques, and we need longitudinal studies to determine if users may find sensing techniques "cool" at first, but later become annoyed by false positives and negatives, for example. However, we strongly believe that such research will ultimately help deliver new and exciting software that fulfills the promise of ubiquitous mobile devices.

ACKNOWLEGEMENTS

We wish to thank Steve Bathiche for the touch sensor used in our prototype, and David Thiel for video production.

REFERENCES

1. Bartlett, J.F., *Rock 'n' Scroll Is Here to Stay*. IEEE Computer Graphics and Applications, May/June 2000.
2. Buxton, W., *Chunking and Phrasing and the Design of Human-Computer Dialogues*, IFIP'86, 475-480.
3. Buxton, W., *Integrating the Periphery and Context*, Graphics Interface '95, 239-246.
4. Compaq, *iPAQ Pocket PC*. 2000.
5. Dey, A., et al, *CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services*, IUI'98,47-54.
6. Fitzmaurice, G., et al., *An Exploration into Supporting Artwork Orientation in the User Interface*, CHI'99.
7. Fitzmaurice, G., *Situated information spaces and spatially aware palmtop computers*,CACM **36**(7), p.39.
8. Fitzmaurice, G., Zhai, S., Chignell, M., *Virtual reality for palmtop computers.* ACM TOIS, **11**(3), 197-218.
9. Fraden, J., *Handbook of Modern Sensors*. 1996: Springer-Verlag.
10. Harrison, B., et al., *Squeeze Me, Hold Me, Tilt Me! An Exploration of Manipulative User Interfaces*, CHI'98.
11. Hinckley, K., Sinclair, M., *Touch-Sensing Input Devices*, CHI'99, 223-230.
12. Horvitz, E., *Principles of Mixed-Initiative User Interfaces*, CHI'99, 1998, 159-166.
13. Horvitz, E., et al., *The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users*, UAI'98, pp. 256-265.
14. Horvitz, E., Jacobs, A., Hovel, D., *Attention-Sensitive Alerting*, UAI '99, pp. 305-313.
15. Pascoe, J., Ryan, N., Morse, D., *Issues in Developing Context-Aware Computing*, HUC'99: Springer-Verlag.
16. Rekimoto, J., *Tilting Operations for Small Screen Interfaces*, UIST'96, 167-168.
17. Rekimoto, J., *Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments*, UIST'97, 1997, 31-39.
18. Saffo, P., *Sensors: The Next Wave of Infotech Innovation*. Institute for the Future: 1997 Ten-Year Forecast. p. 115-122.
19. Sawhney, N., Schmandt, C., *Nomadic Radio: Scaleable and Contextual Notification for Wearable Audio Messaging*, CHI'99, 96-103.
20. Schilit, B., Adams, N., Want, R., *Context-Aware Computing Applications*, Proc. Mobile Computing Systems & Applications, 1994, Santa Cruz, CA: IEEE.
21. Schmidt, A., *Implicit human-computer interaction through context*, 2nd Workshop on Human Computer Interaction with Mobile Devices, 1999, Edinburgh.
22. Schmidt, A., Aidoo, K., Takaluoma, A., Tuomela, U., Van Laerhove, K., Van de Velde, W., *Advanced Interaction in Context*, HUC'99, 89-101.
23. Schmidt, A., et al. *There is more to context than location*, Proc. Int'l Wkshp. on Interactive Applications of Mobile Computing (IMC98), Rostock, Germany.
24. Small, D., Ishii, H., *Design of Spatially Aware Graspable Displays*, CHI'97 Companion, 367-368.
25. Verplaetse, C., *Inertial proprioceptive devices: Self-motion-sensing toys and tools.* IBM Systems Journal, 1996. **35**(3&4): p. 639-650.
26. Want, R., Fishkin, K.P., Gujar, A., Harrison, B.L., *Bridging physical and virtual worlds with electronic tags*, CHI'99, 370 - 377.
27. Weiser, M., *The Computer for the 21st Century*. Scientific American, 1991 (September): p. 94-104.
28. Williams, L., *SmartQuill*, British Telecom Labs, 1997.
29. Zhai, S., Buxton, W., Milgram, P., *The Partial Occlusion Effect: Utilizing Semi-transparency for Human Computer Interaction.* TOCHI **3**(3), 1996.
30. Zhai, S., Morimoto, C., Ihde, S., *Manual and Gaze Input Cascaded (MAGIC) Pointing*, CHI'99, 246-253.

Petitioner Samsung Ex-1054, 0010