# Sensor Synaesthesia: Touch in Motion, and Motion in Touch

**Ken Hinckley**[1], **Hyunyoung Song**[1,2]

[1]Microsoft Research
One Microsoft Way, Redmond WA 98052
kenh@microsoft.com

[2]University of Maryland
Computer Science: College Park, MD 20742
hsong@cs.umd.edu

## ABSTRACT
We explore techniques for hand-held devices that leverage the multimodal combination of touch and motion. Hybrid touch + motion gestures exhibit interaction properties that combine the strengths of multi-touch with those of motion-sensing. This affords *touch-enhanced motion* gestures, such as one-handed zooming by holding one's thumb on the screen while tilting a device. We also consider the reverse perspective, that of *motion-enhanced touch*, which uses motion sensors to probe what happens underneath the surface of touch. Touching the screen induces secondary accelerations and angular velocities in the sensors. For example, our prototype uses motion sensors to distinguish gently swiping a finger on the screen from "drags with a hard onset" to enable more expressive touch interactions.

### Author Keywords
Sensors, touch, gestures, multimodal input, mobile devices

### ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Input

### General Terms
Human Factors

## INTRODUCTION
Touch-screen input is ubiquitous in mobile interaction. But mobile devices also include motion sensors such as accelerometers and gyroscopes to support contextual sensing [19,31], digital photography [23], and games. The widespread availability of mobile devices with both touch and motion-sensing presents an opportunity to explore novel interaction techniques that leverage synergistic roles for *touch and motion* as complementary modalities.

Synaethesia is a secondary sensation, such as color, experienced by some people in response to an actual perception, such as listening to music. In the context of touch input, there is a corresponding *sensor synaesthesia* of accelerations, angular velocities, and torques that mobile devices can perceive coincident to the contacts detected by a touch-screen. These secondary sensor responses may offer hints about touch that enhance the expressiveness of touch-screen interaction. We build on a few prior examples (e.g. [12,17,29]) to show how touch+motion represents a promising design space of synaesthetic gestures that has not been recognized or systematically explored before.

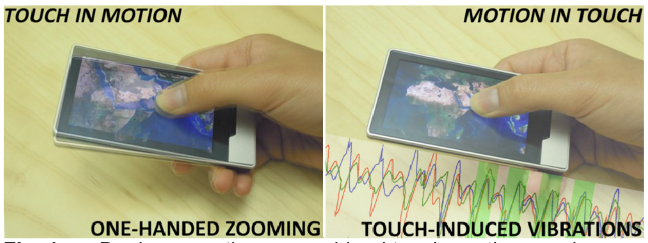

**Fig. 1.** Dual perspective on combined touch+motion sensing.

*Touch-enhanced motion* techniques put "touch in motion" to produce new gestures (*Fig. 1, left*). These gestures can include information from the touch, such as the number of contacts and their positions, as parameters to a motion gesture. For example, the user can tilt the device (motion) to zoom into a specific screen location (touch).

*Motion-enhanced touch* techniques (*Fig. 1, right*) put the "motion in touch" to enable more expressive touch interactions, such as soft vs. hard taps, or gentle swipes vs. drags with a hard onset. These techniques use the incidental vibratory motion induced by finger contact to add nuances to the expression of touch, or to infer context of use (e.g., how the user held the device when he touched the screen).

Collectively these perspectives represent a "hidden dimension of touch" that we seek to explore in the design space of motion sensing techniques that we propose, the novel techniques that we implement, and in the informal user study evaluating these new techniques that follows.

### Synaesthetic Touch + Motion Techniques
To explore the design space of touch + motion, we implemented several techniques on a handheld device.

#### *Touch-Enhanced Motion Techniques*
***1) Tilt-to-zoom*:** The user can zoom in or out relative to a specific point on the screen by holding the tip of the thumb at that point and tilting the device. The user's touch input cues the system to "listen" for motion gestures. This delimits *Tilt-to-zoom* from other incidental movements of the device. This also affords direct control over the center of expansion via the *(x,y)* coordinate indicated by the user's thumb. This offers one-handed continuous zooming, unlike pinch-to-zoom, which requires one hand to hold the device, and the other hand to articulate the pinch gesture.

***2) Pivot-to-lock*:** The user can touch the screen while pivoting the device to a new viewing orientation. This locks the screen at its current landscape/portrait format, for usage scenarios such as browsing the web while lying in bed.

***3) Hold-and-shake*:** The user can touch an icon and shake the device to apply a command, such as *Undo,* to that icon. This integrates selection with the motion gesture itself [17].

***4) Tip-to-select*:** The user can frame a portion of the screen with two thumbs and then tilt the device to select the indicated region. This shows how multi-touch combines with motion to fluidly support additional desirable interactions, without interfering with pinch-to-zoom.

### *Motion-Enhanced Touch Techniques*

The four techniques above illustrate new synaesthetic touch-motion gestures. The following techniques employ motion sensors to lend secondary nuances to touch.

***5) Soft-vs-Hard-Tap*:** Softly tapping an item activates it, but tapping an item harder "drills into it" to allow direct navigation to a secondary target. For example, a soft tap on the calendar tile of our device's home screen navigates to the *day view* showing all appointments for the day, but a hard tap navigates directly to the current appointment.

***6) Swipe-vs-Hard-Drag*:** Gently swiping the screen scrolls or pans the display as usual, but a drag with a hard onset (that is, striking an item and dragging it) allows the user to directly drag and rearrange items in the view, without having to activate an "edit" mode (e.g. via tap-and-hold).

***7) Context of Touch:*** We also explore how accelerometer and gyroscope signals may help infer the context of touches.

## Design Attributes of Touch + Motion

Synaesthetic touch+motion gestures exhibit a number of desirable design properties that collectively offer designers a new implement in the toolbox of mobile interaction. To explore the design space of touch+motion gestures, we carefully considered the design properties of this class of gestures. We then deliberately chose the techniques enumerated above to populate this space of touch+motion gestures with examples that illustrate each property. Here, we briefly discuss these properties, and indicate the corresponding techniques that illustrate each of them.

***State transitions for motion gestures.*** We use *FingerDown* and *FingerUp* events to delimit motion gestures from other movements of a device. This affords chunking and phrasing [4] of touch+motion gestures in a kinesthetic mode that is salient to the user [35]. When motion sensing is used to enhance touch, the onset (or release) of touch indicates when to pull the corresponding motion data out of the sensor stream. Example: All seven techniques listed above.

***Mixed-mode Parameters.*** A touch-screen indicates an (x,y) position in addition to the *FingerDown* and *FingerUp* events. Meanwhile, tilt can apply a continuous control to the indicated location. Example technique: *Tilt-to-zoom*.

***A comfortable and imprecise target to delimit motion.*** Systems must distinguish motion gestures from other incidental handling of a device. A button can serve this role but diverts attention [19] and requires constant muscular strain while moving the device. Some of our touch-motion gestures let the user gently rest a finger anywhere on the screen while moving the device. Such motions demand less attention, do not impose a particular hand-grip, and may be more comfortable to articulate. Example: *Pivot-to-lock*.

***Contextual commands.*** Some motion gestures can use the touch point to indicate specific objects on the screen. This enables motion gestures that apply to a specific icon [17], photo, or region of the screen. Example: *Hold-and-shake* a specific icon to *Delete* an icon, or *Undo* that deletion.

***Multi-touch motion gestures*.** Unlike a physical button, touch allows for multiple points of contact, which can be combined with motion to afford additional interaction states. Example: *Tip-to-select* a region of the screen, as indicated by the bounding rectangle of two thumbs.

***Fluidly interleave touch and motion.*** Some techniques may interleave motion gestures with complementary direct-touch movements. For example, a user can hold his thumb still and tilt-to-zoom, but then slide his thumb to transition directly to panning. Examples: *tilt-to-zoom*; *tip-to-select* interleaves two-thumb zooming with rectangular selection.

***Expressive Touches*.** By leveraging the motion sensors in more of a background-sensing role [19], different nuances of touch can be sensed. This also allows simply resting a finger on an object to be treated differently than a distinct tap on an object, for example. Examples: *Soft-vs-Hard-Tap*, *Swipe-vs-Hard-Drag*, and *Context of Touch* techniques.

## RELATED WORK

Combining motion gestures with touch is a simple idea that has been little explored. Hassan's "Chucking" technique [17] uses a simultaneous touch+motion gesture to toss a file from a mobile device to a wall display. The user holds a finger on the file's icon, while indicating where to place the file via a motion gesture. Rahman [29] also uses touch + motion to measure wrist deflection angles. We contribute novel examples of touch+motion gestures, and we reveal the larger design space of touch+motion by articulating its properties and charting a taxonomy of related techniques.

On digital cameras, sampling device motion at the moment that the user depresses the shutter button can yield enhanced image deblurring [23]. Skinput [16] uses vibratory signals to infer the location of taps on one's forearm. These techniques hint at the strategy of *looking for the motion in touch* that we adopt in some of our techniques.

Several efforts have considered accelerometers as a proxy for pressure input. For example, accelerometer-inferred pressure is not satisfactory for the musical expression of piano-forte on touchscreen-based mobile devices [12], but can support a few discrete states for "expressive typing" on the keys of an accelerometer-enhanced laptop [22].

Motion has also been used as a cue for grip-sensing mobile devices. For example, an accelerometer can trigger implicit grip sensing when a mobile device is held still [24]. Graspables [36] use accelerometers to sense the orientation of objects and to trigger grip sensing at the right moment.

Several efforts consider touch in combination with other modalities. Herot [18] describes a touch screen that senses shear and torque forces. PACER [26] interleaves touch gestures with camera-based motion gestures. Spilling [28] and TouchProjector [2] use direct input on the screen of a device, in combination with movement of the device itself, to augment physically situated interactions. Touch + motion also echoes other examples of synergistic modalities, such as speech + gesture [10] and pen + touch [20].

### Motion Sensing and Delimiters

Many works have explored motion-sensing gestures, as well as background sensing of motion as a contextual cue. As this implies, motions may be explicitly triggered by the user, but motions also occur incidental to handling and transporting a mobile device. While several tradeoffs have been explored (e.g. [1,14,19,21]), a fundamental problem for motion sensing is to either map gestures to motions unlikely to occur by accident, or to explicitly delimit motion gestures from other handling of the device. Indeed, delimiters and state transitions [5] are fundamental building blocks of interactive techniques.

Some motion gestures are distinctive enough to serve as delimiters. TimeTilt employs tapping the back of a device, or jerking it forward and back, to signal state changes [30]. Particular patterns of hard contact forces (such as whacking a device [21]) can provide motion gestures that are fairly robust to false positive recognition errors. However, to avoid false positives, these approaches limit the space of motion gestures. Articulating the delimiters also may slow down some of the resulting interactions.

Many systems instead employ mechanical button presses to delimit motion gestures and manipulations (e.g. [9,37]). This opens up a larger space of permissible motions, but requires that the user find and hold a pushbutton on the device. This may force the user to adopt a particular hand grip, or reduce the range of hand and wrist motions [29] that are comfortable to articulate. Some systems have used *indirect* touch [14,19,24], but few systems have considered direct touch [17,29] to delimit motion gestures.

For manipulative motion gestures, tilt dynamics are often based on rate mappings to compensate for the limited range of motion that is possible when tilting a device [1,14,19]. The exact transfer function and appropriate feedback for such mappings can be tricky to optimize [8,11]. Other systems use larger-scale absolute movements to support a metaphor of direct spatial navigation [2,9,13,28].

### Additional Considerations for Mobile Interaction

One-handed input is widely recognized as a desirable property for mobile interaction (e.g. [17,29]). Yet the ubiquitous pinch-to-zoom gesture requires both hands on mobile devices: one hand holds the device while the other hand pinches. As such, an effective one-handed method for continuous zooming could prove valuable to mobile users.

Attention is a precious resource in mobile interaction [19,21,31]; indeed, limiting the visual demands of interfaces has motivated many motion-sensing techniques. Many of our proposed touch+motion gestures implicitly require some degree of visual attention, since they involve interaction with the touch-screen. Yet our techniques complement existing motion-sensing techniques, and they populate a middle ground of gestures that offload some aspects of what would otherwise be purely touch-based visual interactions onto the motion channel.

## DESIGN SPACE OF MOTION SENSING TECHNIQUES

We can now map out the design space of motion sensing techniques. The following tableau (Fig. 2), informed by previous taxonomies [3,6], delineates the design space by the *property sensed* versus the *activation mechanism*.

| | **INDIRECT** | | **DIRECT** | | |
|---|---|---|---|---|---|
| | **Mech. Buttons** | **Indirect Touch & Pressure** | **Direct Touch** | **None (pure motion)** | |
| **Acceleration (linear accelerometer)** | Virtual Shelves [9] | | ***Pivot-to-lock*** | Auto screen rotation [19] | Absolute (Θ) |
| | Tilt for text [37] | Tilt scrolling [14,19] | ***Tilt-to-zoom*** Measuring wrist angles [29] | Rock'n'Scroll [1] | Relative (ΔΘ) |
| | Expressive Typing [22] | | ***Hard-tap*** ***Hard-drag*** Piano forte [12] | Whack gestures [21] | Hard contact force |
| | | | ***Hold+shake*** ***Tip-to-select*** Chucking [17] | TimeTilt [30] | Motion gesture |
| | | Graspables [36] Grip sensing [24] | | | Stability (no motion) |
| **Angular Velocity** | Deblur image on shutter press (accel. + gyro) [23] | | ***Thumb vs. finger usage*** | ***Tapping corners*** Dual-screen Reader [7] | |
| **Vibratory** | | Scratch input [15] | Skinput [16] | Bone conduction microphones | |
| **Shear, Torque** | | Gummi bendable computer [34] | Vector touch-screen [18] | | |
| **Position-based motion sensing** | Chameleon [13] (Hold button + sensed motion of device) | | TouchProjector [2], Spilling [28], (Direct input on screen + sensed motion of device) | PhoneTouch [32] (motion signal on phone, at same time as phone contact with surface) | |

**Fig. 2.** Design space of motion sensing techniques, organized by the *property sensed* (rows) versus *activation mechanism* (columns).

The property sensed is the primary axis of organization. We classify it as *acceleration* (sensed by linear accelerometer), *angular velocity* (gyroscope), *vibratory* signals, *shear & torque* forces, or *position-based* motion sensing. Within the acceleration category, the rightmost column characterizes the techniques based on how they use the accelerometer.

The columns of the table classify techniques by the *activation mechanism* employed to trigger an interactive response to sensed motions. *Indirect* activation mechanisms include mechanical buttons as well as touch, pressure, or grip sensors integrated with the device. *Direct* activation mechanisms include direct-touch contact with the display, as well as "none" (that is, no activation mechanism), for techniques that employ always-active motion sensing.

We populate each cell of the resulting matrix with references to representative systems and techniques. The new techniques described in this paper are shown in **bold**.

The taxonomy of Fig. 2 situates our techniques within a constellation of related motion-sensing techniques, but does not emphasize mixed-mode mappings for touch + motion gestures. Thus, we enumerate our techniques in a second table (*Fig. 3*) that shows combinations we have explored.

| **TECHNIQUE** | **State Transitions** | **Touch Parameter** | **Motion Parameter** |
|---|---|---|---|
| **Tilt-to-zoom** | **2** (Hold) | **(x,y) point** for focus of expansion | Change in forward-back tilt angle ($\Delta\Theta_x$) |
| **Pivot-to-lock** | **2** (Hold) | **Entire screen** (touch anywhere) | Absolute rotation ($\Theta_z$) |
| **Rotate object** (variant of Pivot-to-lock) | **2** (Hold) | **(x,y) point** for center of rotation | Change in rotation ($\Delta\Theta_z$) |
| **Hold-and-shake** | **2** (Hold) | **Select object** or icon | Gesture: three spikes in ($\Theta_{xy}$) |
| **Tip-to-select** | **2** (Two-finger Hold) | **Rectangle** (two touch points) | Gesture: one quick jerk in ($\Theta_{xy}$) |
| **Hard-tap** | **0→2→0** (Hard contact on rapid FingerDown-FingerUp event sequence) | **Select object** or icon | Hard contact force: Spike in ($\Theta_z$) |
| **Hard-drag** | **0→2** (Hard contact on FingerDown event, with no FingerUp) | **Select object** + **drag path**. | Hard contact force: Spike in ($\Theta_z$) |
| **Tap-and-hold** (e.g. for context menus) | **0→2** (No sensed motion and no dragging after FingerDown) | **Select object** or icon | Stability (no motion) |

**Fig. 3.** Mixed-modality mappings explored by our techniques.

The *state transitions* show how we incorporate the motion signal into the interactive technique, in terms of Buxton's 3-state model [5], where state 0 is *out-of-range* (no finger contact) and state 2 is dragging (finger on screen).

The *touch parameter* is the aspect of the touch signal that the technique leverages, which can be any of the following:

- **Entire screen**: a touch anywhere on the screen.
- **(x,y) point**: the centroid of the touch point.
- **Drag path**: the path followed by the finger.
- **Select object**: the user touches a specific icon, object, or region of the screen to select it.
- **Rectangle** (two touch points): a bounding box.
- **Contact / Shape**: the geometry of the touch(es).

Our techniques currently do not include any examples of *Contact / Shape* parameters for touch+motion gestures, as our prototyping devices do not report this information. Nonetheless these parameters suggest paths for future work.

The *motion parameter* specifies the contribution of the motion signal to the mixed-mode parameters of a gesture. These use the same nomenclature as the attributes from the rightmost column of Fig. 2; we also note which axes of the accelerometer each technique employs.

## IMPLEMENTATION

We implemented our techniques as 3rd party applications for Windows Phone 7 (WP7). We also implemented some techniques on a Zune HD. Both devices have capacitive multi-touch screens and an integrated three-axis linear accelerometer. WP7 supports sampling the accelerometer at 30Hz, whereas the Zune HD supports 60Hz. We used both platforms because of limited availability of WP7 devices, and to user-test *Tilt-to-zoom* at the 60Hz sampling rate.

We have implemented a WP7 photo browser which includes all of our techniques except Tilt-to-zoom; this remains a separate application. However, we believe that a well-considered implementation could support all of our gestures simultaneously (using approaches suggested by [1,19,30], for example). Here we focus on optimizing the individual techniques to assess their value to users.

### Motivation for Application Scenarios & Techniques

Several of our techniques are motivated by pain-points that we have observed in current mobile user interfaces. These include the inability to articulate continuous one-handed zooming, the difficulty of browsing the web while lying in bed due to undesired automatic screen rotation, and the tedium of performing a succession of taps to drill down to commonly used options or views. We also intentionally chose examples that populate our design space of touch+motion gestures, so that each technique illustrates specific design properties. Most of the techniques were implemented in a WP7 photo browser; we chose this as a demonstration application because all of the touch+motion gestures of interest have useful semantics in the context of browsing, organizing, cropping, and viewing individual photos or collections of photos [8].

## TOUCH-ENHANCED MOTION TECHNIQUES

### Tilt-to-Zoom

As noted earlier, pinch-to-zoom is difficult to perform, one-handed. As such, touch-screen devices often use double-tap for one-handed zooming, but double-tap only supports discrete zooming. Furthermore, in the context of zooming within a web browser, the first of the two taps is prone to accidentally activating links. This forces the user to hunt for "empty" space, or text that is not hyperlinked, to successfully zoom via double-tap. Thus, a reliable and effective way to zoom with one hand is desirable for mobile devices. Because one-handed continuous zooming would open up many mobile usage scenarios, such as a user referring to a map while encumbered with shopping bags or holding a child's hand, such a technique complements pinch-to-zoom even if it is not necessarily faster.

The user zooms into a point on the screen by holding his thumb at the desired location, and tilting the device. Tilt-to-zoom thus affords directly zooming into locations such as the corners of a web page. Pinch-to-zoom cannot directly express this because there is not room to expand two fingers at the edges of the screen. Yet our technique is compatible with pinch-to-zoom; a device can support both techniques. This lets the user employ the approach that best suits his current usage context.

#### *Implementation Issues & Considerations*

We implemented Tilt-to-zoom in a simple map navigation application. Touching down a finger on the screen starts an ambiguous "pan-or-zoom" mode. We say the mode is ambiguous because at the moment of the *FingerDown* event, the system does not yet know if the user will pan or zoom the display. Sliding the thumb beyond a minimum distance threshold triggers the standard panning behavior. If the user instead tilts the device while the finger remains still, this triggers tilt-to-zoom. Lifting the finger from the screen exits the mode. The user can also hold the thumb still at the end of a panning motion, and then tilt, to transition directly to zooming. Our technique therefore fluidly interleaves both panning and zooming.

Tilting the top edge of the screen *away from yourself* is mapped to zoom-out, while tilting the top edge of the screen *closer to yourself* is mapped to zoom-in. In early pilot testing, we found that slightly more than half of users preferred this mapping, but some users with a different mental model preferred the opposite mapping. The zooming speed is a rate-based function of the *change* in forward-back tilt angle [14,19]. The zooming speed is therefore determined *relative to the angle at which the user starts touching the device*: that is, whenever the user touches the screen, this resets the "resting" orientation (where the speed of zooming is zero) to the current forward-back tilt angle. We chose a transfer function with a rate of expansion that seemed "about right" in early pilot tests, balancing rapid zooming against the desire for fine control of the zoom.

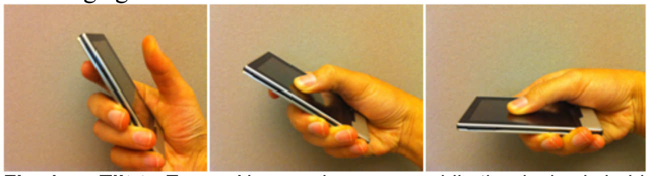
**Fig. 1. Tilt-to-Zoom:** No zooming occurs while the device is held normally (left). Touching the screen enters the zooming mode (middle). Zooming is then mapped to the change in tilt angle (right).

To avoid accidentally triggering tilt-to-zoom with small motions, the user must pass through a dead-band of ±5° around the starting orientation before zooming starts. Zooming uses tilt along the device's forward-back axis, which corresponds to ulnar/radial deviation of the wrist [29] for a typical single-handed grip (Fig. 1). However, it is possible to tilt the device without significant deviation of the wrist by adopting a precision grip on the device, and using the fingertips to subtly tilt it forward and back like a miniature seesaw. The rate-based transfer function enables zooming a long distance with a small change in the tilt.

### *Visual Affordance for Motion Gestures*

There has been little exploration of visual languages for motion gestures [25] that show the desired direction or degree of tilting required by an interaction. To convey that tilt-to-zoom requires forward-back tilting, we provide a *perspective box* near the borders of the map that appears axis-aligned at the start of a tilt motion, but becomes perspective-distorted as the user tilts further from the starting angle. The vanishing point appears to be either somewhere above the screen, when zooming in, or somewhere below the screen, when zooming out. This conveys the current zoom direction as well as the tilt axis that responds to zooming. Our pilot testing suggested that the perspective box enforces the metaphor, while providing feedback of the correct direction to tilt the device.

## Pivot-to-Lock

Automatic screen rotation [19] is commonplace on mobile devices, but in some usage contexts the technique can lead to undesired changes in the screen orientation. Attempting to read a web page while lying down on a couch, or in bed, are common examples that many of us have experienced. To address this, the Apple iPad and iPhone include a global screen lock (a hardware button for the iPad, and a software setting for the iPhone) to disable automatic screen rotation, but this makes it tedious to change orientation. The *MobileRSS* iPhone app (tinyurl.com/yb37rac) pops up an icon when the user reorients the screen. Users can touch this icon to toggle between the *lock* and *automatic* screen rotation option. This icon appears for two seconds every single time the user reorients the device. This may distract the user, and it interferes with reading the underlying text.

### *Implementation Issues & Considerations*

The interaction metaphor of *Pivot-to-Lock* is that the user can "pin down" the screen to keep it at its current viewing orientation. That is, holding a finger on the screen while pivoting the device to a new viewing orientation locks the screen at its current landscape or portrait format.

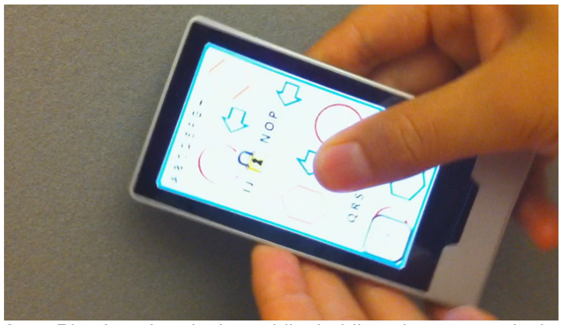
**Fig. 2.** Pivoting the device while holding the screen locks the current viewing orientation. A small padlock icon provides feedback.

For this technique, the location of the touch point does not matter. Thus it is an example of a touch-motion gesture that uses the imprecision of a touch anywhere on the entire screen as a way to delimit motion-sensing gestures from other incidental movements of the device. Although this is the only example of such a gesture that we implemented, it is clear that motion-sensing gestures for other "global commands" that act on the entire screen (or on the currently active application) could be added, so long as the required motions do not conflict with one another.

We implement standard automatic rotation to any of the four canonical screen orientations, but we check to see if the user is holding a finger on the display before we respond. Like the *Tilt-to-zoom* technique, on a *FingerDown* event *Pivot-to-lock* starts in an ambiguous mode. If the user touches down and slides his finger, this again triggers panning. If the device rotates substantially with little movement of the finger, we turn off the panning interpretation. When the device has rotated far enough to represent a new viewing orientation, we trigger *Pivot-to-lock* instead of automatically rotating the screen.

Visual feedback in the form of a padlock icon (🔒) appears to indicate that the screen has entered the lock state. The screen remains locked after the user lifts his finger. The obvious design question is how the user can later deactivate the lock. To prevent the screen lock from becoming a heavyweight mode, we opted to leave the lock on as long as the user remains at the current viewing orientation. As soon as the user rotates the device to a new orientation (without pinning down the screen), this turns off the lock and restores the automatic screen rotation functionality. This solution ensures that the user cannot become "trapped" in a mode that they do not know how to escape.

### Tip-to-Select

Selecting or clipping content from the current screen view can be problematic for touch-only interfaces, if only because many of the obvious input events and state transitions available to the interaction designer must be assigned to other, higher-priority tasks. For example, to select a region of text on the Apple iPad, the user must tap and then manipulate small blue selection handles to indicate the desired passage. Indicating a rectangular selection by framing it with two fingers is an oft-demonstrated technique in the literature, but in typical mobile device interfaces, two-finger manipulation is already consumed by the pinch-to-zoom gesture.

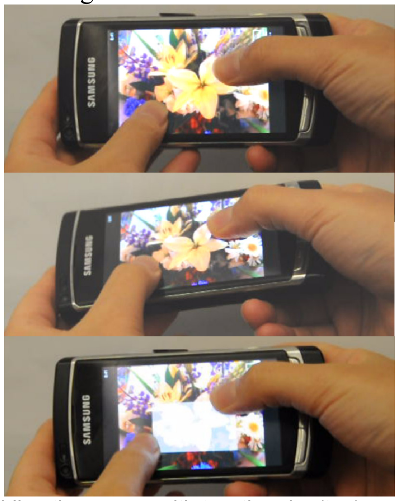

**Fig. 3.** Holding the screen with two thumbs (top) and then tipping the device away (middle) selects the indicated region (bottom).

#### *Implementation Issues & Considerations*

Our photo browser supports two-fingered zooming on individual photographs. However, with the *Tip-to-Select* technique, if the user holds both fingers stationary and quickly tips the device away and back, this triggers a state transition to the *selection mode*. The user can then use the two fingers to directly manipulate the opposite corners of a marquee selection (*Fig. 3*). When the user lifts his fingers, this crops the selected region from the photo.

Note that *Tip-to-Select* fluidly interleaves two-fingered zooming with the selection mode. The user can start by zooming to the desired portion of a photograph, for example, and then keep the fingers in contact with the display while tipping the device to trigger selection.

*Tip-to-Select* responds to a change in orientation from the start of the motion signal. The user must quickly depart from the current orientation by at least ±15°to trigger the transition to selection mode. The user can tip by this amount in any direction. The user then returns to the original orientation within a short time-out. Hence, *Tip-to-Select* represents a gesture that is recognized as a whole, rather than continuously interpreted as a direct manipulation. The motion component of *Tip-to-Select* is related to the jerk gesture employed by TimeTilt [30].

### Hold-and-Shake

Context menus (e.g. for photos, icons) can be difficult to access on a mobile device. Bringing up the menu via tap-and-hold is a standard solution. The user must search through the menu for the desired option, and then tap on it via hand-eye coordination. Hence context menus are slow to activate, and demand significant visual attention.

On iPhone and Android devices, shaking applies to the entire application (e.g. undo, shuffle playlist). However, a problem with shaking as a pure-motion gesture is that it is prone to unintended activation. As a result, a system must be careful what it accepts as "shaking," which makes it more difficult to articulate the gesture when it is intended.

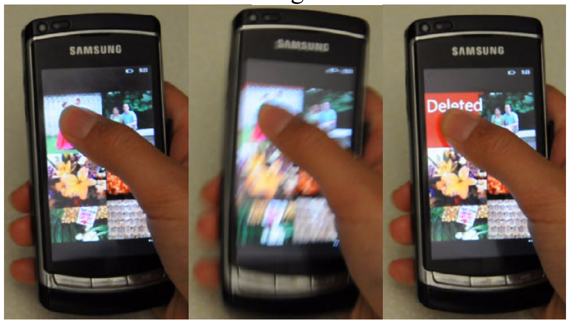

**Fig. 4.** Holding a photo (left) while shaking the device (middle) deletes the photo (right). Shaking again undoes the deletion.

#### *Implementation Issues & Considerations*

We instead explore *Hold-and-Shake* as a contextual touch+motion gesture. This gesture simultaneously demonstrates two interaction properties of interest. First, it illustrates how holding touch indicates a mode switch from the "neutral" mode, where incidental motions can be ignored, to a "gesture" mode where the system looks for motion gestures. Second, *Hold-and-Shake* illustrates how this transition to gesture mode can be integrated with object selection, by selecting the underlying object or icon that falls underneath the (x,y) coordinate of the touch. Clearly the concept of *touching an object to integrate mode switching with object selection* [20] could be applied to other contextual motion gestures, such as holding a photo and moving the device in various directions [17,30].

The user can hold a photo and shake the device to delete the indicated photo. Instead of asking the user to confirm whether the photo should be deleted, our prototype just replaces the photo with a red square that says *Deleted*. The user can later undo the operation by holding on the former location of the photo and again shaking the device to *Undo* the action. This leads to a nice symmetry between the original action and the gesture needed to reverse its effects.

Like our other gestures, *Hold-and-Shake* starts in an ambiguous mode. For example, touching down but then sliding the finger on the screen scrolls the list of photos, and deactivates motion-sensing mode. If the user instead touches down while the finger does not slide, we then recognize the shaking gesture by a heuristic that looks for three successive peaks in the motion signal (within a 2s time interval). Once motion starts, but before the shaking gesture has completed, we relax the distance threshold (for the transition from holding to scrolling) because the touch contact point may move while the user shakes the device.

Note that the standard tap-and-hold interaction can still be supported in the presence of *Hold-and-Shake*. The system just recasts tap-and-hold as a touch that occurs while the device is not sensing significant motion.

## MOTION-ENHANCED TOUCH TECHNIQUES

The techniques detailed in the sections above focus on touch-enhanced motion gestures. We now turn our attention to the complementary perspective of techniques that sense motion consequent to touching a device's screen. We implement *Soft-vs-Hard-Tap* and *Swipe-vs-Hard-Drag* to demonstrate this approach. We conclude this section with some further possibilities for augmenting touch with gyroscope + accelerometer motion sensing.

### Soft-vs-Hard-Tap

Touch interfaces for small-screen devices often tradeoff breadth for depth: fewer options are presented on any one screen, so that more taps are required to navigate to a particular option. To address this, we implement *Hard-vs-Soft-Tap* in a mock-up of the WP7 home screen "tiles" and a Calendar application with *daily agenda* and *current appointment* views. This exhibits the depth-first nature typical of navigational structures on mobile phones. The home screen offers a context where making the interactions less tedious (and more fun) seems particularly appropriate.

By introducing a distinction between soft-taps and hard-taps, and the design rule that *two soft taps equals one hard tap,* our technique allows users to selectively jump ahead: softly tapping an item activates it, but tapping an item more forcefully "drills into" it to allow direct navigation to a secondary target. For example, in our prototype, a soft-tap on the calendar tile in the device's home screen navigates to the *day view* with all appointments for the day, but a hard tap navigates directly to the current appointment.

This technique also enlivens interaction with the home screen, while providing feedback for hard vs. soft taps, by animating the tile that the user taps in proportion to the force exerted. Lightly tapping an icon subtly expands the tile to show that it has been activated, but striking it more forcefully causes it to animate with more of a "splat."

We distinguish hard vs. soft taps by looking at the difference between the accelerometer peaks in the first 200ms of the touch. If the separation is larger than 0.3g, it is a hard tap. This heuristic could be improved by considering the force of the touch in relation to ongoing background movements, such as walking [21].

### Swipe-vs-Hard-Drag

Many views on mobile devices can be scrolled or panned, but at the same time, the user may want to rearrange the items in the view by standard drag-and-drop actions. Examples of such views include home screen icons, photos in an album, or web bookmarks. The problem is that making a swipe motion to scroll conflicts with drag-and-drop, so the system typically must introduce a special "edit" mode to support rearranging items.

Our technique offers a new way to finesse this problem by using motion sensing to distinguish gentle swipes from hard drags. In our prototype, gentle swipes scroll through a list of photos. Hard drags "grab" the photo that the user lands on, to drag it to a new position. The user thus has direct access to both scrolling and dragging. This technique is similar to *Soft-vs-Hard-Tap*, except that the finger remains in contact with the display and specifies the dragging path.

### Further Possibilities: Accelerometers, Gyros, & Touch

We have fewer example techniques for *motion-enhanced touch*, but we believe this perspective is equally important. Our devices only include integrated accelerometers, which limits their sensitivity to some types of motions [23].

Users employ various hand grips on mobiles, such as interacting with a thumb, versus the index finger of the opposite hand. If we can sense such nuances, they may form valuable contextual cues that can be used to fine-tune the handling of the touch input signal itself. For example, the software might adjust interaction thresholds to make scrolling a web page with a thumb more forgiving. Or, a soft-keyboard might be able to optimize its input model if it knows whether a given tap is a thumb, or an index finger.

To explore such possibilities, we attached an external gyroscope to a Zune HD and produced sensor traces. As shown below (*Fig. 5*), performing vertical swipes on the screen with a *Thumb,* versus a *Finger,* results in distinct motion-sensor signals. We found the gyro signal (top trace) more sensitive to these types of subtle motions than the accelerometer. Furthermore the gyro senses motion in the reference frame of the device, as opposed to the world coordinate system of the accelerometer, thus making it easier to recognize the device-centric motions of interest.

This suggests that such contextual inferences could be made with appropriate recognition techniques, and a sufficient corpus of training data. We implemented a simple user-specific recognizer as a proof of concept, but more work must be done to generalize and validate this. For example, the gyroscope and accelerometer have complementary properties that can be combined to help enhance recognition (e.g. by unifying the sensors into a 6DOF inertial reference frame via the *direct cosine matrix* [27] or other algorithms).

We also tried recognizing finger taps on the inert rear casing of the device. Our video illustrates the recognizer responding to the user's finger tapping the case at the top-left corner, and the top-right corner. Tapping each corner produces a distinct motion which we recognize as a different type of tap (e.g. for forward/back gestures in a web browser). Thus, the gyro and accelerometer signals can be used to localize "touch" interactions to some degree, without any true touch sensing being used at all [15,16].

Our recognizer uses a typical training-data approach. We manually register several templates for the desired motion. Template matching is performed on a candidate temporal data sequence. If the sum of Euclidean distances between the input values and the template is within an error threshold, the recognizer reports a match.

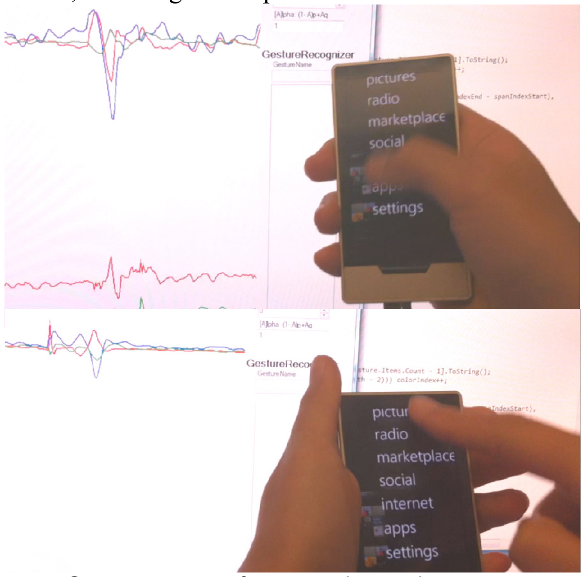

**Fig. 5.** Sensor traces from a three-axis gyroscope and accelerometer for one-handed scrolling with the thumb (top) versus two-handed scrolling with the index finger (bottom). See also video.

While we believe these are intriguing possibilities, they nonetheless remain to be adequately explored and validated by future work. For example, the motion signals coincident with touch interactions also might form valuable sources of contextual data for probabilistic interaction frameworks, such as that proposed by Schwarz et al. [33].

## INFORMAL EVALUATION & REACTIONS FROM USERS

In addition to our early pilot studies which helped to iteratively refine the initial implementation of our techniques, we conducted an informal evaluation to garner feedback on our techniques from test users. The goal of the evaluation was to assess each technique's value proposition, as well as its strengths and weaknesses, with test users. We also sought ideas for possible enhancements by observing users' experiences, as well as their comments.

### Participants

We recruited 10 Microsoft volunteers (3 female, 7 male, ages 22-55, none affiliated with our team). Each received $10 for participation. Nine participants were touch screen phone users (5 iPhone, 3 Windows Mobile, 1 Android), and one participant was a generic flip style mobile phone user.

### Procedure

The experimenter demonstrated each of the six primary touch+motion gestures discussed above. Participants then tried each feature, and spent about 10 minutes using each one. For the Tilt-to-Zoom gesture, users tried both the WP7 prototype (which offered the *perspective box* feedback) and the Zune HD prototype (which offered 60Hz sampling rate for more responsive tilt control, but no perspective box).

After trying each technique, users filled out 7-point Likert scale questions, and we asked users what the best and worst thing was about each technique. After trying all six techniques, users rank-ordered them for overall preference.

### Results

Overall, the Likert questions revealed positive to strongly positive responses to memory load, learning curve, and ease of control. Many users commented that the features were simple, fluent, seamless, or intuitive. The general area that indicated the most concern by users was ease of discovery, which tended towards neutral to mildly positive responses on the Likert scale questions. Although our current focus is not on self-revelation mechanisms for motion gestures [25], this does suggest we should improve discoverability. In general, this is a little explored topic in motion sensing.

#### *Learning Touch+Motion Gestures*

The physical metaphors that many of our techniques exploit appeared to help users learn the techniques, while also making the interaction memorable. For example, one user commented that *Pivot-to-lock* "mimics a natural technique to prevent things from moving." Another user described *Hold-and-shake* as a "salt and pepper" metaphor. *Hard-tap* and *Hard-drag* correspond to real-world experiences with striking objects: as one user stated, "the splat animation is a natural mapping between gesture and the resulting action."

Although all users found the gestures easy to learn, we observed that in the early going, in general there are four steps that most users progress through:

**(1)** Learn the device reacts to motion. One user stated "I'm not accustomed to shaking the phone," while another's initial reaction was "tilting may not be natural." However, most users had seen or used motion-sensing before.

**(2)** Holding the screen starts listening to motion. Because of the limited screen real estate, users had a tendency not to touch the screen unless there was a clear purpose. (One user stated: "I don't keep my finger on the screen if there isn't a target," while another said that "at first, holding a finger against the screen was unintuitive"). However, most users had no problem holding the screen while moving.

**(3)** Users must generate motion in synchrony with touching the screen. With Tilt-to-zoom, some users made anticipatory tilting motions before their finger actually touched down on the screen, which made the technique seem less responsive.

**(4)** The user must learn how tilt is mapped to the interface control. This was easy for direct angular mappings (e.g. *Pivot-to-Lock, Tip-to-Select*) but took some getting used to for the rate-based *Tilt-to-Zoom*. Our rate-based zooming function appeared to be a bit over-sensitive, leading to overshoot, and hence should be further refined [11]. A couple of users suggested the alternative of imparting discrete zooming steps by making "whip" tilting motions. Nonetheless, despite these limitations, *Tilt-to-zoom* (and *Pivot-to-lock*) resonated very strongly with most test-users.

### *Semantics Matter*

It is important to map touch+motion gestures to suitable interaction semantics. There were several users who did not like *Hard-tap*, but reacted positively to *Hard-drag*, which is based on a very similar motion. These users commented that if they make a mistake with *Hard-tap*, they first have to realize this, and then figure out how to go back. On the other hand, if the user intends to do *Hard-drag* but scrolls a bit by mistake instead, the cost of this error is essentially zero, and the user can just try hard-dragging the item again.

We observed a similar issue with *Pivot-to-lock*. All users liked being able to lock the screen orientation, but several users wanted the lock to be persistent (for example, one who really liked *Pivot-to-lock* also stated that "auto unlock is a horrible feature"). Several users wanted the device to remain at a fixed screen orientation if they showed the screen to someone else, or passed the device to that person.

### *Lightweight Interactions*

Users appreciated that they could use movements of the device for *Pivot-to-lock* and *Tilt-to-zoom*, or intensity of the movement for *Hard-tap* and *Hard-drag*, to address common problems that they experience in mobile device interaction. Compared to tedious sequences of time-outs or touch gestures, users liked how they could combine wrist, finger, and device motion to articulate lightweight interactions with less "friction" in the user interface. For example, users commented that the techniques were "intuitive and easy to transition to different modes," "easy and magical," or that "the icons are alive!"

### *Physical Effort and Ergonomics*

On the other hand, for motion based interaction to be useful, the interactions must carefully consider appropriate grips as well as the wrist's range-of-motion [29]. For example, although we designed *Pivot-to-lock* with the idea that users would "pinch" the device between their fingers, and spin the device (somewhat like a propeller), many users performed *Pivot-to-lock* by adopting a rigid grip on the device while using ulnar/radial deviation of the wrist to turn the device on its side. This is somewhat uncomfortable as the wrist has limited mobility along this axis. However, in a more naturalistic usage context, where the user actually lies down in a bed while rigidly gripping the device, no wrist motion would be required. Nonetheless, this leads us to consider teaching users the appropriate grip on the device as part of the self-revelation strategy for our gestures.

Likewise, the simplistic recognition heuristics that we currently use for some of our gestures, such as *Hard-tap* and *Hard-drag* ("Hard tapping is too hard on my fingers"), or *Hold-and-shake* ("I had to shake vigorously"), currently requires the user to make more forceful or effortful motions than perhaps is necessary.

### *Overall Preference for the Techniques*

Users' preference rankings for the techniques reflected the everyday problems that they experience using their mobile phones. The need to support effective one-handed zooming, as well as to avoid undesired automatic screen rotation, strongly resonated with many of the users. Five of the ten users explicitly mentioned "one-handed" interaction as the best thing about *Tilt-to-zoom*. Another user loved the fact that the center of expansion is controllable, unlike pinching. Users of the iPhone in particular felt that having a simple gesture to lock the screen orientation was valuable. Several users particularly appreciated shortcuts for common interactions ("avoiding unnecessary taps is always nice"). *Hold-and-shake* was the only technique that nobody chose as their favorite technique (or runner up), largely because shaking took too much effort, and took too much time. However, because our study only includes 10 subjects, keep in mind that the difference between the rankings is not significant (Kruskal-Wallis test, $\chi^2_{(5)}$=16.74, p>0.05).

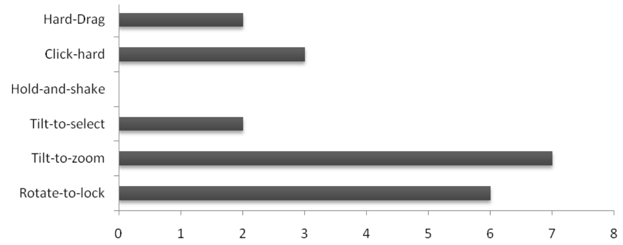

**Fig. 6.** Number of participants who chose each technique as their favorite technique, or as the runner-up, in their rank orderings.

## CONCLUSION AND FUTURE WORK

Our interactions are designed to take unique advantage of the properties of handheld mobile devices, while the user is interacting with the device at hand. However, because of this, it is unclear how touch+motion techniques would translate to other form-factors, such as slate devices, or to other usage contexts, such as employing a device while it rests flat on a desk. For example, the motions for *Tilt-to-zoom*, *Pivot-to-lock*, and *Tip-to-select* cannot be applied when the device is resting flat on a surface such as a desk.

In our current prototype, we deliberately picked interactions that work well with an accelerometer. However, integrating accelerometers with gyros, proximity sensors, or vibratory and shear/torque sensors could greatly enrich the vocabulary of motion at our disposal. For example, combining accelerometers and gyros into an inertial reference frame enables each sensor to make up for shortcomings of the other, which could help to improve our existing techniques, or open up possibilities for others.

We have shown how the multimodal combination of *touch-enhanced motion* enables new types of motion gestures such as holding the device and tilting it to zoom; pivoting the device to a new orientation to lock the screen rotation; framing a portion of the screen with two thumbs and then tilt the device to select the indicated region; or holding an icon on the screen and shaking the device to apply a command to that specific icon.

We have also explored *motion-enhanced touch* techniques that tease out further nuances of touch when the user strikes the screen. Sensors such as accelerometers and gyros can pick up the subtle ripples in the motion stream that spread out from the point of impact. We have implemented hard-tapping and hard-dragging as examples of this perspective, while also presenting sensor traces that suggest further contextual sensing techniques may be possible.

Collectively, all of these techniques help to reveal the hidden dimension of touch. Direct interaction is about much more than just the touch modality itself. This work provides one example of how touch interactions can be extended and enriched by considering the interplay of touch with other modalities, including touch + motion sensing.

**ACKNOWLEDGEMENTS**
We thank Andy Wilson, Hrvoje Benko, Daniel Wigdor, and Michel Pahud as well as our test users and the reviewers.

**REFERENCES**

1. Bartlett, J.F., Rock 'n' Scroll Is Here to Stay. IEEE Computer Graphics and Applications, 2000(May/June 2000): p. 40-45.
2. Boring, S., Baur, D., Butz, A., Gustafson, S., Baudisch, P. Touch Projector: Mobile Interaction Through Video. *CHI'10*.
3. Buxton, W., Lexical and Pragmatic Considerations of Input Structure. Computer Graphics, 1983. **17**(1): p. 31-37.
4. Buxton, W. Chunking and Phrasing and the Design of Human-Computer Dialogues. *IFIP Information Processing `86*. Amsterdam: North Holland Publishers.
5. Buxton, W. Three-state model of graphical input. *Human-computer interaction - INTERACT'90*. Amsterdam: Elsevier Science Publishers B. V. (North-Holland).
6. Card, S., Mackinlay, J., Robertson, G. The Design Space of Input Devices. *CHI'90*.
7. Chen, N., et al. Navigation Techniques for Dual-Display E-Book Readers. *CHI'08*.
8. Cho, Murray-Smith, & Kim. Multi-Context Photo Browsing on Mobile Devices Based on Tilt Dynamics. *Mobile HCI'07*.
9. Chun Yat Li, F., Dearman, D., Truong, K. Virtual Shelves: Interactions with Orientation-Aware Devices. *UIST'09*.
10. Cohen, P.R., Dalrymple, M., Moran, D.B., Pereira, F.C., Sullivan, J.W. Synergistic use of direct manipulation and natural language. *CHI'89*.
11. Eslambolchilar, P., Murray-Smith, R. Tilt-Based Automatic Zooming and Scaling in Mobile Devices. *Mobile HCI'04*.
12. Essl, G., Rohs, M., Kratz, S. Use the Force (or something) - Pressure and Pressure-Like Input for Mobile Music Performance. *NIME 2010 Conf. on New Interfaces for Musical Expression*.
13. Fitzmaurice, G., Buxton, W. The Chameleon: spatially aware palmtop computers. *CHI'94*.
14. Harrison, B.L., Fishkin, K.P., Gujar, A., Mochon, C., Want, R. Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. *CHI'98*.
15. Harrison, C., Hudson, S. Scratch Input: Creating Large, Inexpensive, Unpowered and Mobile Finger Input Surfaces. *UIST'08*.
16. Harrison, C., Tan, C., Morris, D. Skinput: Appropriating the Body as an Input Surface. *CHI'10*.
17. Hassan, N., Rahman, M., Irani, P., Graham, P. Chucking: A One-Handed Document Sharing Technique. *INTERACT'09*.
18. Herot, C., Weinzapfel, G., One-Point Touch Input of Vector Information from Computer Displays. Computer Graphics, 1978. **12**(3): p. 210-216.
19. Hinckley, K., Pierce, J., Sinclair, M., Horvitz, E. Sensing techniques for mobile interaction. *UIST 2000*.
20. Hinckley, K., et al. Manual Deskterity: An Exploration of Simultaneous Pen + Touch Direct Input. *CHI 2010 Ext. Abstracts*.
21. Hudson, S., Harrison, C., Harrison, B., LaMarca, A. Whack Gestures: Inexact and Inattentive Interaction with Mobile Devices. *TEI 2010 Conf. Tangible & Embedded Interaction*.
22. Iwasaki, K., Miyaki, T., Rekimoto, J. Expressive Typing: A New Way to Sense Typing Pressure and Its Applications. *CHI'09*.
23. Joshi, N., Kang, S.B., Zitnik, C.L., Szeliski, R. Image Deblurring using Inertial Measurement Sensors. *Proc. SIGGRAPH 2010*.
24. Kim, K.-E., et al. Hand Grip Pattern Recognition for Mobile User Interfaces. *Proceedings of AAAI/IAAI-2006: Innovative Applications of Artificial Intelligence*. 2006.
25. Kratz, S., Ballagas, R. Unravelling Seams: Improving Mobile Gesture Recognition with Visual Feedback Techniques. *CHI'09*.
26. Liao, C., Liu, Q., Liew, B., Wilcox, L. PACER: Fine-Grained Interactive Paper via Camera-Touch Hybrid Gestures on a Cell Phone. *CHI'10*.
27. Mahoney, R., Nonlinear Complementary Filters on the Special Orthogonal Group. IEEE Trans. on Automatic Control, 2008. **53**(5): p. 1203-18.
28. Olsen, D.R., Clement, J., Pace, A. Spilling: Expanding Hand-held Interaction to Touch Table Displays. *IEEE TableTop '07*.
29. Rahman, M., Gustafson, S., Irani, P., Subramanian, S. Tilt Techniques: Investigating the Dexterity of Wrist-Based Input. *CHI'09*.
30. Roudaut, A., Baglioni, M., Lecolinet, E. TimeTilt: Using Sensor-Based Gestures to Travel through Multiple Applications on a Mobile Device. *Interact '09*.
31. Schmidt, A., et al. Advanced interaction in context. *Handheld and Ubiquitous Computing (HUC'99)*.
32. Schmidt, D., Chehimi, F., Rukzio, E., Gellersen, H. PhoneTouch: A Technique for Direct Phone Interaction on Surfaces. *UIST'10*.
33. Schwarz, J., Hudson, S., Mankoff, J., Wilson, A.D. A framework for robust and flexible handling of inputs with uncertainty. *UIST'10*.
34. Schwesig, C., Poupyrev, I., Mori, E. Gummi: A Bendable Computer. *CHI'04*.
35. Sellen, A., Kurtenbach, G., Buxton, W., The prevention of mode errors through sensory feedback. Human Computer Interaction, 1992. **7**(2): p. 141-164.
36. Taylor, B., Bove Jr., V. Graspables: Grasp-Recognition as a User Interface. *CHI'09*.
37. Wigdor, D., Balakrishnan, R. TiltText: Using tilt for text input to mobile phones. *UIST'03*.