I Sense a Disturbance in the Force: Mobile Device Interaction with Force Sensing

James Scott, Lorna M Brown and Mike Molloy
Microsoft Research Cambridge
7 JJ Thomson Ave, Cambridge CB3 0FB, UK
{jws, lornab, v-mimoll}@microsoft.com

ABSTRACT
We propose a new type of input for mobile devices by sensing forces such as twisting and bending applied by users. Deformation of the devices is not necessary for such "force gestures" to be detectable. Our prototype implementation augments an ultra-mobile PC (UMPC) to detect twisting and bending forces. We detail example interactions using these forces, employing twisting to perform application switching (alt-tab) and interpreting bending as page-down/up. By providing visual feedback related to the force type applied, e.g. of an application window twisting in 3D to reveal another and of pages bending across, these force-based interactions are made easy to learn and use. We also present a user study exploring users' abilities to apply forces to various degrees, and draw implications from this study for future force-based interfaces.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces – Input devices & strategies.

General terms: Measurement, Performance, Design, Experimentation, Human Factors

Keywords: Force, sensors, mobile devices, UMPC

INTRODUCTION

For mobile computing devices such as tablet or slate PCs, PDAs, or new smart phones such as iPhone, much of the surface of the device is devoted to the screen. The trend for increasing screen sizes is continuing, since larger screens facilitate information presentation. However, since users also want their devices to be small, less and less space is available for physical controls such as numeric or alphanumeric keys, or buttons, jog dials, etc. While these devices typically have touch screens, the use of the portion of the touch screen for a button interface reduces the available screen area for information presentation.

In contrast, the sensing of physical forces can provide an input mechanism for devices without requiring external surface area on the device. By sensing forces such as bending or twisting that users can apply to the casing itself, input can be provided to the device's operating system or applications. The device itself does not need to actually bend for forces to be detectable, so this sensing mechanism is compatible with today's rigid devices.

In this paper, we describe the concept of force sensing as a user input and compare its qualities with other input mechanisms. We detail our prototype implementation based on an ultra-mobile PC (UMPC) using thin load sensors, and algorithms to determine which forces have been applied using a combination of inputs from the sensors. We present some interaction techniques where we use applied forces to replace shortcut keys, in particular twisting forces are used to indicate switching between applications (alt-tab), and bending forces are used to indicate flipping through pages of a document (page-down/up). We then describe our user study concerning the fundamental capabilities of users to apply such forces, looking at the number of force levels that can be reliably produced for both twisting and bending interactions.

RELATED WORK

In the past decade there has been a move to consider new interaction techniques for mobile devices, away from the traditional use of either physical or virtual buttons, or on screen interactions. Much of this work has used accelerometers to sense tilt, e.g. Rekimoto [11] and Hinckley *et.al.* [5], while more recent work has also used inertial sensing to sense the way in which a user shakes a device [14]. One disadvantage of tilting as an input mechanism is that it can make it harder to read the screen if it is tilted away (the final example above avoids this problem by providing all information through audio and tactile stimuli). Applying physical forces to the device will not require tilt and, therefore, could be a more practicable option.

The Gummi concept and prototype by Schwesig *et al.* [12] proposes a vision for a fully flexible mobile computer. The elastic deformation of such a device can be used to interact with it, e.g. to zoom in and out by bending it like a lens. Our work differs in that users do not actually bend the device significantly when they apply forces to it; users may find it more or less difficult to use than a device which actually bends or articulates. It is also much simpler to integrate force sensing with current device designs which incorporate many rigid components.

Some haptic force feedback devices have been created which sense forces applied by a user as well as output forces to the user. An example of this is the haptic rotary knob (Snibbe *et al.*) [13], which can sense the forces ap-

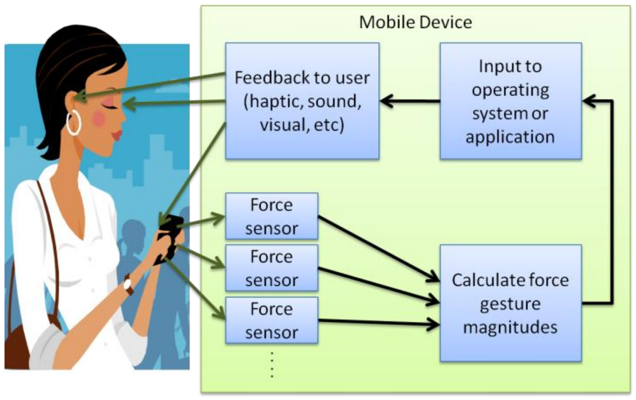

Mobile Device
Feedback to user (haptic, sound, visual, etc)
Input to operating system or application
Force sensor
Force sensor
Force sensor
Calculate force gesture magnitudes

Figure 1: Overview of force-based interaction system

plied by a user. In contrast to our rigid device, this device rotates when the user applies a force. Such devices do not tend to have been used in mobile devices, and sensing the forces applied by the user is normally not the main focus of such work. More relevant to our work is the Haptic media controller by Maclean *et al.* (2002) [8] which uses orthogonal force sensing (using strain gauges) to sense forces applied to different faces of a thumb wheel in addition to rotation of the wheel to provide richer interactions.

Much of the previous work on force sensing in HCI only uses direct pressure, whereas our work uses several different types of forces (e.g. bending and twisting). For example, researchers at Xerox PARC used pressure sensors [3] in order to detect which side a user was gripping a device to inform "handedness-aware applications". Another difference between this and our work is that we propose the use of force sensing for active interactions rather than this more passive use of the technology. Another example of the use of direct pressure, but this time using active interactions, is the sensing of the pressure of a pen or finger on a touch screen, e.g. Mizobuchi *et al.* [9]. Using pressure on the screen can obscure the screen; our system uses forces applied to the body of the device and thus avoids obscuring the screen. Jeong *et al.* [6] used direct pressure applied to a force sensor mounted on a 3D mouse to control movement speed in virtual environments.

Blowable user interfaces [10] where the user blows on areas of a screen, are another recent development. As with force sensing, the user does not obscure the screen they are looking at. While blowable interfaces benefit from hands-free operation, force sensing benefits from not requiring the user's face to be near the controlled device.

SENSING USER-APPLIED FORCES

Users can apply many types of forces to devices, and these can be sensed and used as input for applications or the operating system running on such devices, with graphical, audible, haptic or other types of feedback provided to the user, as shown in Figure 1. Such forces can be detected with a variety of sensors including load sensors which sense direct mechanical compression between parts of the casing, strain gauges which sense the elastic stretching of the case due to the force applied, or pressure sensors sensing gaseous or liquid pressure in a channel in the device which expands or contracts to a small degree as the user applies forces to the external casing of the device.

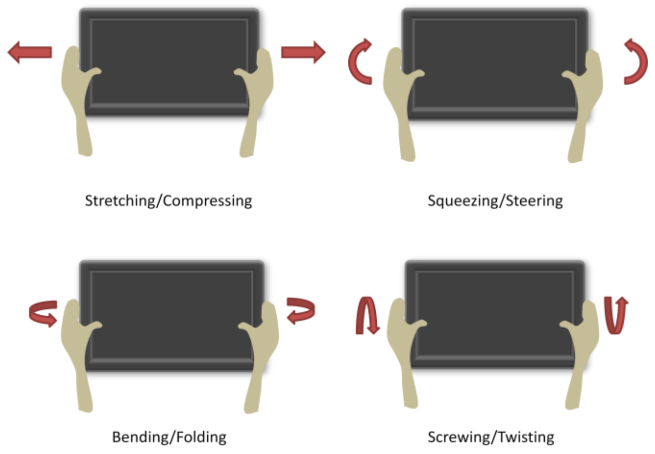


Figure 2: Four user-applied force gestures that can be sensed

Users can apply various categories of forces, with four examples shown in Figure 2. Each force can be applied in two directions (the way indicated by the red arrows and the opposite directions), and can be applied to a variable degree. Furthermore, forces can be applied along different axes (e.g. vertically as well as horizontally in Figure 2) and to different parts of a device. Thus, force sensing can potentially be a rich source of input information and is not limited to the simple presence or absence of force.

Note that, unlike with flexible technologies such as Gummi, detecting force does not require that the device must actually bend significantly, or be articulated around a joint. The device can be essentially rigid, with only very small elastic deformations due to the applied pressure. This is crucial as it allows force sensing to be deployed in many types of device which rely on current-day components such as LCD screens or circuit boards which are rigid, and avoids the need to use less mature or more expensive flexible technologies in a device.

Qualities of Force Sensing Input

Force sensing has some intrinsic qualities that make it an interesting alternative to other forms of input. With force sensing, the user interacts with the casing of the device, turning an otherwise passive component that just holds the device together into an active input surface. Forces applied to the casing are mechanically transmitted through it and parts attached to it. Therefore, unlike for physical controls such as keys or dials, force sensors do not necessarily need to be located at the external surface of the device in the part of the device that the user holds, and device cases can be made with fewer holes for physical switches, which makes for more robust, more easily manufactured, and reduced form factor devices. Force sensing shares this advantage with accelerometer-based input and similar sensors.

Another advantage of force sensing over other physical controls is the ability to apply an input in many places or ways. A user can employ one hand or two, and/or another physical object such as a desk, in order to generate forces. Users have an inherent and intuitive understanding of how

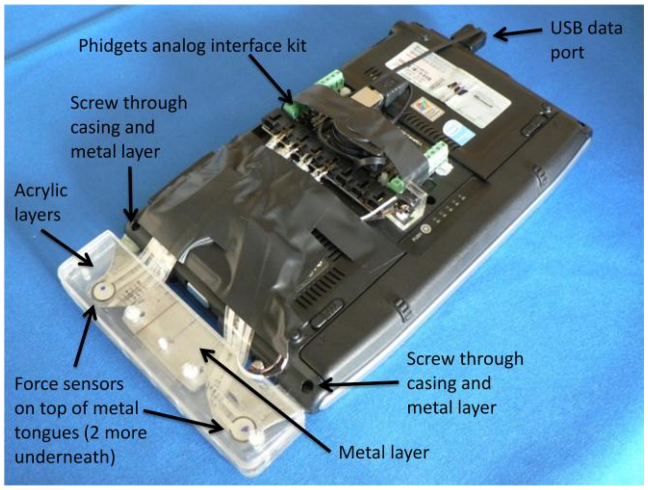

USB data port
Phidgets analog interface kit
Screw through casing and metal layer
Acrylic layers
Screw through casing and metal layer
Force sensors on top of metal tongues (2 more underneath)
Metal layer

Figure 3: Force sensing prototype using augmented UMPC, rear view showing added hardware

to apply forces by analogy to other tasks in the real world such as snapping off a piece of chocolate, wringing a towel, etc. When physical controls are present on a device, their location leads to orientations and ways of holding the device that are more useful or less useful, limiting the flexibility of the device for use in different applications and use scenarios.

We can also compare force sensing with other types of input such as touch or multi-touch input on the screen, the use of accelerometers or other physical sensors, etc. Compared to shaking or tilting the device or using a touch screen, force sensing allows the screen to be kept at the most natural viewing angle and unobscured. During force sensing input the device can be essentially stationary, thus inputs can be made subtly in situations where that is useful (such as during a meeting).

However, when comparing these options with force sensing, the most important point is that it is not an either-or decision as many input mechanisms can be built in to the same device. Mobile computers are general-purpose tools with many applications, and thus the inputs required are complex. By employing multiple input mechanisms with separate functionalities, we can avoid overloading any one input mechanism. The best choice of input mechanism at any given time will depend on the application, the user, the usage scenario, the environment/context of use, and so on. Thus, we present force sensing as a mechanism to add to the richness of input mechanisms on a device rather than a way of replacing another mechanism.

PROTOTYPE FORCE SENSING HARDWARE

We built a prototype force sensing device by augmenting a Samsung Q1 UMPC with a custom additional casing inside which were placed four force sensors, as shown in Figure 3. We used an additional casing rather than an integrated solution for feasibility reasons; UMPCs and other mobile computers are tightly packed with components so it would be difficult to incorporate force sensing hardware without redesigning the case.

The additional casing comprises an exterior acrylic section made of various layers cut with a laser cutter, screwed together and hand-filed to round the corners. This is placed around a central hand-cut metal layer (magnesium alloy) which is attached to the UMPC by the same screw points that normally hold the top and bottom half of the UMPC casing together. The metal layer is therefore sandwiched tightly between the top and bottom of the UMPC casing (which were trimmed to make space for the metal to stick out) and is rigidly attached to the UMPC. The acrylic casing is not directly attached to the UMPC itself, it is attached rigidly to the metal layer by a tight friction fit and two screws to prevent slip.

For the force sensors we used four FlexiForce 0-25lb load sensors from TekScan Inc. These are small (the sensing area is 10mm wide and can be made smaller) and thin (0.2mm) making them simple to incorporate into the prototype, and in future work to incorporate into a redesigned device casing. The force sensors are placed tightly between “tongues” on the metal layer and the acrylic layers (using some blobs of epoxy to make sure of a good fit), two on each side of the metal layer, at the top and bottom. The placement of the force sensors (and shape of the tongues and of the casing in general) is dictated by the forces that we intended to sense; this prototype is intended to sense “twist” and “bend” gestures (as shown in Figure 2).

The Flexiforce sensors lower their electrical resistance from over 30MΩ when no pressure is applied to around 200kΩ when squeezed hard between a thumb and finger. We apply a voltage to one contact of each Flexiforce sensor and measure the voltage at the other, with a 1MΩ pull-down resistor. These voltages are measured using the analog to digital converters in a Phidgets[1] Analog Interface Kit which is attached to the back of the UMPC (as shown in Figure 3) and software on the UMPC queries the force values through a USB interface.

Determining Force Gestures Using Sensors

With Flexiforce sensors (as compared to other types of force sensor such as strain gauges or pressure sensors), we can detect only compression forces. We could, of course, preload a sensor so we can detect release from compression, or even load a sensor to halfway to detect both compression and release of compression. However, initial experiments showed preloading to be unreliable since friction causes the “zero” point to move each time the device is manipulated. Therefore, the design above was intended to sense only compression.

When building the casing described above we had intended to sense each of the four gestures (twisting and bending in two directions each) using the sum of inputs from two force sensors. Twisting in each direction should compress two diagonally opposite sensors, while bending should compress either the two front sensors or the two back sensors. The initial gesture magnitude calculations were therefore done according to the following rules:

[1] http://www.phidgets.com/products.php?product_id=1018

TwistClockwise = TopFront + BottomBack

TwistAnticlockwise = TopBack + BottomFront

BendTowards = TopBack + BottomBack

BendAway = TopFront + BottomFront

where *TwistClockwise* etc are the gesture magnitudes and *TopFront* etc indicate the raw sensor values (which rise with increased pressure). However, this did not work as well as expected.

After some experimentation, we found that pressure was concentrated at the edges of the metal tongues and not, as we had hoped, centred on the tongues where we had placed the force sensors. We derived a more optimal positioning whereby the top tongue's two sensors were placed at the side edge of the tongue, while the bottom tongue's two sensors were placed at the top and bottom sides of the tongue. This meant that the top tongue's sensors reliably detected bending gestures, while the bottom tongue's sensors reliably detected twisting gestures.

TwistClockwise = BottomBack

TwistAnticlockwise = BottomFront

BendTowards = TopBack

BendAway = TopFront

While this reliably senses gestures being applied, one problem with this is that the sensors also sometimes trigger when the wrong gesture is applied, e.g. the *BottomFront* sensor indicating *TwistAnticlockwise* would sometimes trigger for *BendAway*. It is easy to see why by examining the first set of rules above; despite the off-centre placing, two sensors are still sometimes triggered by a single action due to the force on the tongues as a whole, depending on the exact angle of the force applied by the user. We therefore came up with a third set of rules, which avoided such false positives:

TwistClockwise = BottomBack - TopBack

TwistAnticlockwise = BottomFront - TopFront

BendTowards = TopBack - BottomFront

BendAway = TopFront - BottomBack

Each sensor is the primary sensor for one gesture but is sometimes active for another gesture, and conversely each gesture has one primary sensor but sometimes causes another sensor to trigger. These rules avoid the false positive cases by regarding a sensor as correctly indicating its primary gesture to be active when it is active and the primary sensor corresponding to the gesture for which it is sometimes active is NOT active.

In order to normalise the forces required by the user to apply different gestures, we built a self-calibrating system which tracks the maximum gesture value (calculated as above) and a threshold value, and normalises the gesture magnitude between these two levels. The threshold value is set to be the maximum gesture value while another gesture is currently being applied by the user at at least half its maximum magnitude. In other words, while the user is applying a gesture, we detect jitter levels for all the other gestures and ignore such jitter even when the device is idle. Sensing threshold levels while users apply gestures works well because (a) we are sure that the value is spurious (since the user is applying a different gesture), and (b) during the application of different gestures, more jitter occurs than when idle, thus the resulting thresholds are conservative for the idle state.

Both the gesture sensing and normalisation algorithms have proven to work well in practice. We have built two augmented UMPCs for redundancy, each has slightly different sensor placements and hence sensitivities due to small physical differences (the prototypes were hand-built), yet the same algorithm self-calibrates well on both devices.

The UMPCs were used for an internal demo event at Microsoft where hundreds of novice users tried it. Many were able to use the device with no explicit instruction (by watching it being used by someone else, or reading instructions on a poster), and nearly all users were able to operate it with a few seconds of coaching.

One problem that we did encounter was that the maximum levels were raised by the occasional high-arm-strength user, and then low-arm-strength users had difficulty. This can obviously be addressed by manual or automatic recalibration, by regarding the maximum as a personalised setting (like mouse sensitivity), or by setting a static maximum for a device with well-known characteristics.

We explore usability by novice users further in our user study presented later in this paper.

INTERACTION USING FORCE SENSING

Force sensing is a general input mechanism providing a number of scalar values that can be mapped onto any desired functionality. However, as for all input mechanisms, force sensing will be more useful in some cases and less useful in others. In this paper, we focus on one class of interactions, that of replacing shortcut keys on a mobile device which lacks a keyboard (or has a fold-out or detachable keyboard which is not "out").

Page-Down and Alt-Tab Force-Based Interactions

We implemented two force-based interactions on the UMPC. The bend action is used to indicate "page down", and the twist action is used to change foreground application (i.e. "alt-tab" under Windows). With both gestures we provide an accompanying visual feedback which mimics a real-life interaction with the application window or document in question, as shown in Figure 4 and Figure 5.

For page-down, we provide visual feedback as if the user were flicking through a book (by bending the book and allowing pages to flick across). Thus, right hand page appears to flip up and over to the left hand side. For the inverse action, page-up, we use the inverse force, i.e. the user bends the device as if to see the backs of their hands.

For alt-tab, we twist the window vertically from one side of the screen to the other, revealing the virtual reverse side of the window, which is the next window. The windows are kept in a persistent order (unlike alt-tab which puts least-recently-used first), so a user can go forwards or backwards

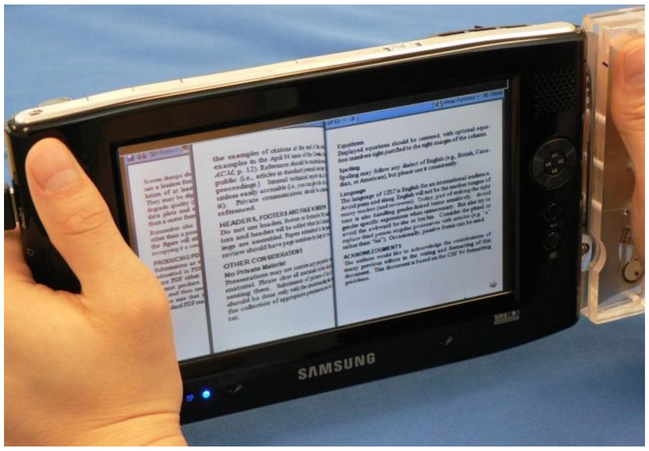


Figure 4: Bending force used to control page down/up, with visual feedback

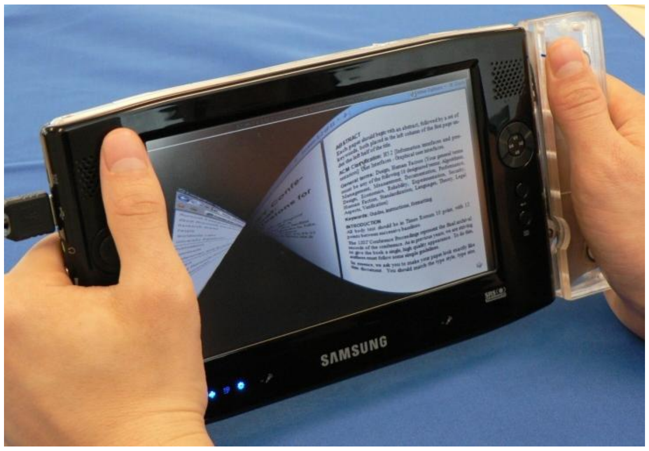


Figure 5: Twisting force used to control application switching (alt-tab), with visual feedback

from the window they are on (and receive differing visual feedback, with the twist matching the force they apply to the device).

For prototyping purposes we implemented these gestures as mock-ups rather than integrating them into a running system. They are implemented using C# with .NET 3.5 and the Windows Presentation Foundation (WPF) library and we use Windows Vista as the UMPC's operating system. A single 3D polygon mesh is used for alt-tab, while two meshes are used for the page-down visualisation (the current page and the next page). We have since integrated the alt-tab gesture with Windows Vista so that twisting causes a screenshot of the current and next applications to be captured; the animation is then activated using those images.

For both interactions, we have currently implemented the interaction by changing a single page/application at a time, and the user can control how far along the animation the system goes by applying a harder or softer force. If the user applies sufficient force to move to the end of the animation (when the new page/application is fully revealed) this is committed and as the force returns to zero the new page/application will continue to be visible. We used 75% of the maximum force as the endpoint of the animation, though of course this can be configured differently. If the user releases the force before they have made the animation reach the end, then the animation goes back to the beginning as the user releases the force, without any persistent change, i.e. the same page/application is in view.

Advantages of Force Sensing for Shortcut Keys

The use of force sensing to replace shortcut keys plays to a number of the strengths of force sensing. It allows mobile devices to avoid using up valuable form factor space on keys and therefore to be smaller, and it turns the passive casing into an active input thus maximising the utility of the whole surface area of the device. The visual feedback presented in response to the forces applied has strong physical analogies with the applied forces, thus making force-based interaction intuitive to discover through observation, and easy to remember and to use.

While devices vary considerably in form factor and the location of control keys on the device, force sensing, if widely deployed, could provide a standard interface for certain control functionalities, since every handheld device must be supported by at least one hand during use.

Finally, the use of force sensing to replace common shortcut keys leaves the touch screen itself free for other types of interaction, thus reducing the need for more complex touch "gestures" that may be required if touch was used as the sole input medium.

Further Work in Force-Based Interaction

Aside from visual feedback, the current prototype provides a click sound when enough force is applied to reach the end of the animation and lock in the new view. We plan to incorporate richer audio cues in future, e.g. cues which also match the physical action taken such as a crumpling sound or page-flicking sound. We also plan to look at the effects of haptic feedback in combination with visual and/or audio feedback, again to explore the analogy with physical movements.

Another modification to be explored in future is that, instead of different force levels being used to move through the animation of a single change, different force levels can indicate levels of change or rates of change. For example, the bend force could cause pages to flip over at a slow rate if applied weakly (such that one page could be flipped at a time), or at a quick rate if a strong force was applied, giving the effect of rapidly flipping through a book.

However, before we studied these further, we first felt it to be appropriate to conduct a more fundamental study of users' abilities with respect to force inputs, which is the subject of the rest of this paper.

USER STUDY

We conducted a user study to assess the capabilities of force sensing as an input method using our prototype. The study aimed to assess the speed at which users could acquire targets of different widths and different distances from a zero-force start position, and to compare performance across the two gesture types (bend and twist).

Participants

Twenty participants were recruited from within our research lab for a between groups study with two conditions:

in the Bend condition users used the bend gesture to interact with the device, while in the Twist condition they used the twist gesture. Ten participants (5 male, 5 female) were assigned to the Twist condition and ten (6 male, 4 female) to the Bend condition. 80% of the participants were aged 25-35, the remaining participants divided between the following age groups: 20-24 (1 participant), 36-40 (1 participant) and 50-55 (2 participants). All but one of the participants were right handed. Around half of the participants were researchers and the others came from a range of job roles, including IT support, software engineers, human resources and marketing.

While conducting the tests, participants were provided with an office chair (with adjustable arms) and desk and could choose to sit in any comfortable way. Some participants leaned on the desk, some sat back in the chair, and some changed positions during the trial. Participants were each given a box of chocolates worth around £5 as a gratuity.

Methodology

Each user trial consisted of a familiarization phase, a training phase, and an experimental phase. After the first and second phases, device calibration was undertaken allowing the user to choose their preferred maximum force values.

In the familiarization phase, we explained the operation of the device, demonstrated the force gesture being studied, and asked the users to experiment with applying forces for a period lasting a minimum of two minutes. Whilst applying forces, users received feedback in the form of a visual "force cursor", representing the magnitude of the force applied, moving along a horizontal bar. At zero force, the cursor was in the middle of the bar, and the user moved the cursor in either direction by twisting/bending one way or the other.

During the calibration that followed, users configured the device with chosen maximum forces by applying a comfortable but hard force in each direction a minimum of three times. We considered the maximum force a user to be comfortable with to be an individual personalization setting (akin to mouse sensitivity).

Both the training and test phases involved the same overall structure of blocks of tests, with the training phase simply allowing the user to become accustomed to the type of tests being applied. This method was based on the discrete task variation of the Fitts' Law task [7]. A block of tests involved a set number of targets on the screen uniformly filling the space between zero force and the maximum configured force in each direction (i.e. for "two target" tests there were actually two targets in each direction). Figure 6 illustrates a test with 4 targets on each side.

For each test, the user was first made to keep the device at zero force for two seconds which left the cursor at the home position in the centre of the screen. Then, a single target was highlighted in purple and the user was asked to move the force cursor as quickly as possible into the highlighted target, and then to hold the force cursor inside the target for 2 seconds. To aid the user, the currently-aimed-at target was highlighted with a yellow glow. After the target

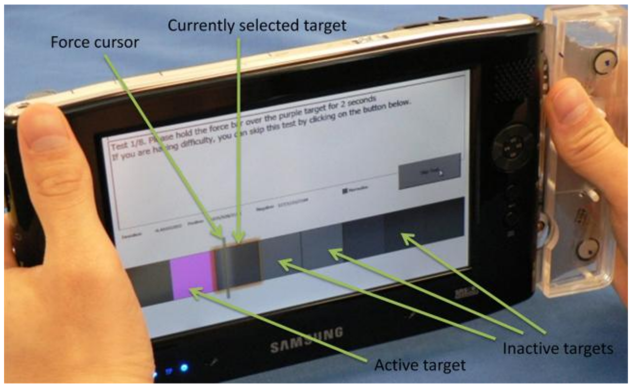

Force cursor
Currently selected target
Test 1/8. Please hold the force bar over the purple target for 2 seconds
If you are having difficulty, you can skip this test by clicking on the button below.
Inactive targets
Active target

Figure 6: Screenshot of software for user trials

was acquired (i.e. the force cursor was kept inside the target for 2 continuous seconds), the purple marking disappeared and the user was instructed on screen to return the force to zero before the next target appeared.

We chose deliberately not to include a button for users to click on a target as soon as they enter it, for three main reasons. First, because force sensing potentially users gripping fingers in both hands, so adding a button would add a significant new factor. Second, with force sensing users can change the position of their hands on the device, so it was not clear where a button should be placed. Third, we wanted to explore button-free interaction which is one of the affordances of force sensing input.

In a given block of tests, users had to acquire each target precisely once, though the targets were presented in random order during a block. During a block of tests the targets were presented immediately one after the other (with a minimum 2-second zero-force period in between). After each block of tests was completed, the user was offered the chance to take a break, and had to press a touch screen button to continue to the next block.

During the training phase there were four blocks with two, four, six and then eight targets in each direction, allowing the user to get accustomed through progressively harder tasks. After the training phase, the user was given the opportunity to recalibrate the force maximums if they had discovered during training that they were set too low (making the system too responsive) or too high (making it hard to get to the furthest targets).

During the main experiment phase, the number of targets in a given block was varied between two and eight targets in each direction, with blocks for each number of targets appearing three times, once during each of three cycles. During the cycles, the order of number of targets was randomized. In total, each participant was asked to acquire 210 targets during the experiment phase. The total duration of all phases of the trial was around 50 minutes.

In case of major difficulty in acquiring a target, after 10 seconds a "skip" touch screen button became available for the user along with on-screen instructions that they could skip the target if they were finding it too difficult.

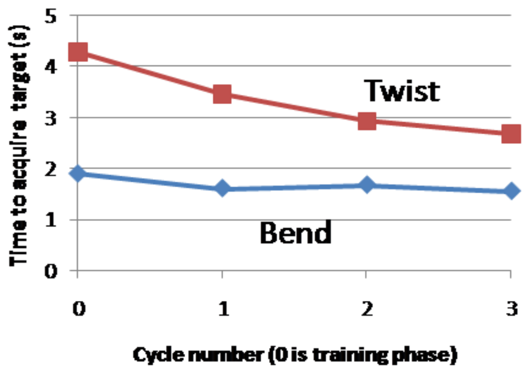


Figure 7: Effect of participant experience on average target acquisition time

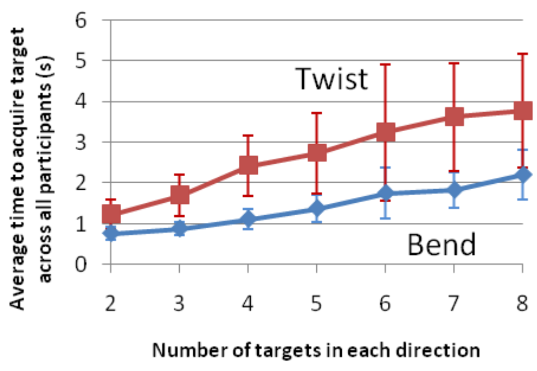


Figure 8: Effect of different numbers of targets on average target acquisition time

At the end of the experiment participants were asked to complete NASA TLX Workload questionnaires [4] so we could assess the perceived workload in each condition.

Hypotheses

We identified four concrete hypotheses to test, in addition to more general findings from the study:

H1. There will be no significant difference in performance between Bend and Twist

H2. There will be no significant difference in perceived workload between Bend and Twist

H3. Target acquisition time will increase as the number of targets is increased

H4. Target acquisition time will increase as the distance to target is increased

Hypothesis 1 and 2 deal with the comparison between the two force gestures in terms of performance and perceived workload. Hypotheses 3 and 4 are based on Fitts' Law [2] which states that target acquisition time will increase as target width decreases (which occurs when number of targets is increased) and distance to target increases.

RESULTS FROM THE USER STUDY

In all the times presented below, the final two seconds are not included, i.e. the time is reported until the target is entered by the force cursor at the beginning of the two continuous seconds required for the test to be complete.

One participant's data has been excluded from the Bend condition as the prototype broke during the trial and it had to be aborted. The prototype was subsequently repaired before further trials. Therefore the data reported herein is only for the remaining nine participants.

Analysis of the data for the Twist condition showed outlier behaviour for one participant (average times more than two standard deviations away from the mean for the majority of the targets). Therefore, this data has been excluded to allow analysis of the behavior of the non-outlying participants. At the same time, we must conclude that some users may have difficulty using a force sensing system and may be more comfortable with another input mechanism.

We do not present or analyse data from the training phase except where specially noted. While we offered users the ability to skip targets, in a total of 3780 experiment-phase targets presented to the 18 non-excluded participants, only 10 were skipped (0.3%).

In this section we present the raw results and statistical analysis, discussing their implications in the next section.

Learning effects and effects of gesture type

Figure 7 shows the effect of the cycle on the average target acquisition time (cycle 0 is the training phase). An ANOVA (analysis of variance) showed a significant effect of cycle (F(3,4545)=10.33, p<0.01). Post-hoc Tukey tests showed that for Twist there was a significant difference (p<0.05) between the training phase and all other phases, and between cycle 1 and 3. However, no significant improvements in performance were observed in the Bend condition, even from training to the experiment itself.

It is possible that the twist gesture was less familiar to participants than the bend gesture in the context of mobile devices (bending is used, for example, to open a clamshell phone), so twisting required more learning for users.

The mean target acquisition time (across all targets) was 1.6 seconds for Bend and 3 seconds for Twist. A between groups ANOVA showed that the average time was significantly faster for the bend gesture F(1,3769)=115.07, p<0.01). The NASA TLX workload results showed that participants' perceived workload was significantly higher in the Twist condition than in the Bend condition (F(1,3769)=2789.75, p<0.01).

Effect of number of targets/target width

Figure 8 shows the effect of number of targets on the average target acquisition time, with error bars indicating the standard deviation for variation between participants. Since targets in a given block of tests filled the whole force bar, the number of targets is inversely proportional to the width of each target. Note that in the figures and discussion, we describe the number of targets *in each direction*, i.e. there are twice as many targets on the screen.

The graph shows that, for both conditions, the average target acquisition time increased as the number of targets increased (and width decreased). ANOVAs showed significant effects of number of targets on performance for both conditions (for Bend, F(6,1889) = 25.42, p<0.01 and for Twist, F(6,1879) = 10.09, p<0.01). Post hoc Tukey tests showed significant (p<0.05) differences between numbers of targets as follows.

For Bend:

- 2, 3 and 4 were significantly faster than 6, 7 and 8
- 5 was significantly faster than 7,8
- 6 and 7 were significantly faster than 8

For Twist:

- 2 was significantly faster than 6, 7 and 8
- 3 and 4 were both significantly faster than 7 and 8
- 5 was significantly faster than 8

Effect of target position

Figure 9 plots the effect of target position, direction, and number of targets in each direction on target acquisition time. For the Bend condition, overall, the time taken to acquire a target decreased as the distance increased (although it can be seen that this effect is less when fewer targets are present). It can be observed that the most difficult target to acquire was the closest target to the left of the zero force position. This is borne out by the analysis. Individual ANOVAs were performed for each number of targets and showed that, for every number of targets except 3 and 4 targets, there was a significant effect of target position ($p<0.05$). Post hoc Tukey tests showed that this was due almost entirely to the first target to the left being significantly slower to acquire than almost all other targets.

A similar pattern is observed for the Twist condition. The first target to the left is again the hardest to acquire.. The analysis confirms this: individual ANOVAs for each number of targets showed a significant effect of target position ($p<0.05$) for all numbers of targets except when there were only 2 targets. Post hoc Tukey tests again revealed that, apart from a few other isolated cases, this result was due to the first target to the left taking significantly longer to acquire than almost all other targets.

We also observe an increase in the time taken for the furthest target to the left in the Twist condition, especially when 5-8 targets are present. However, ANOVA tests found no significant difference between these points and any other positions. Nonetheless it is interesting to note that there is a slight difference between bend and twist for the highest-force targets. Bending remains at the same level of ease, while twisting hard enough to reach the furthest targets seems more difficult, and this was borne out by participants' responses when asked which targets were hardest.

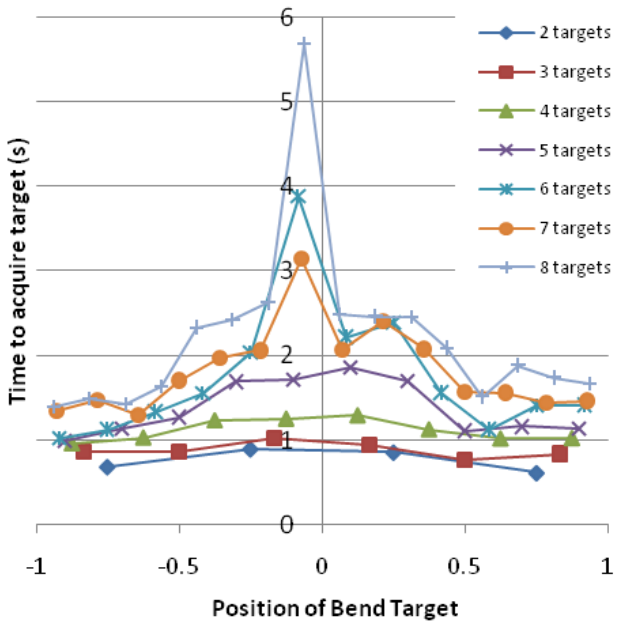
Other factors

An ANOVA showed a significant effect of gender (F(1,3769)=27.79, $p<0.01$). Post hoc Tukey tests revealed that female participants were significantly faster ($p<0.01$) than male participants in the Twist condition. There was no significant difference in the Bend condition. Analysis on the basis of age has not been carried out since 80% of participants fell into the age group 25-35, and analysis on handedness has not been carried out since we had only one left-handed participant.

DISCUSSION

We started our analysis with four hypotheses and these are now considered in turn.

Hypothesis 1 stated that the difference in performance between bend and twist would not be significant. Our data shows that target acquisition is significantly faster in the Bend condition than in the Twist condition, therefore Hypothesis 1 can be rejected. The Bend gesture was learned faster, and also performed better throughout the experiment. There are a number of possible causes of this difference. Despite designing for stiffness, our prototype deforms noticeably during bending but much less so for twisting, which may make it easier for users to apply force in the direction that our force sensors are aligned with, i.e. the deformation acts as a guide. This deformation also meant that users may have avoided setting maximum bend force too high for fear of breaking the prototype (as one did!), thus making the "far" targets simpler to reach. Twisting performance improved significantly during the experiment,



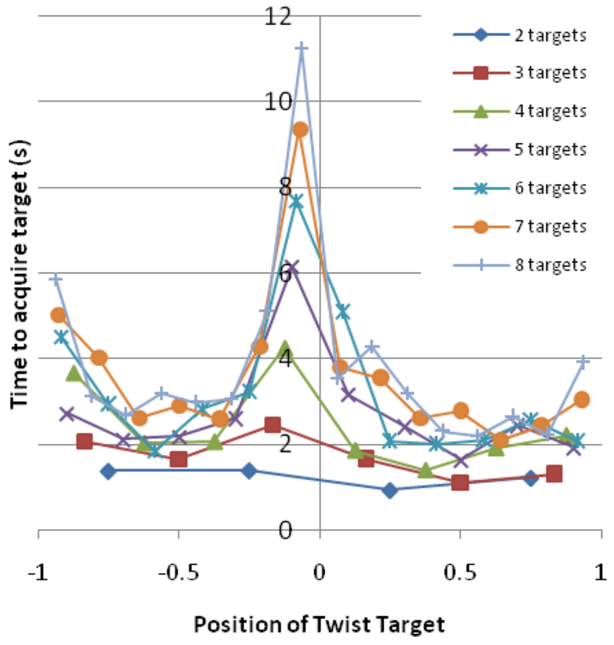


Figure 9: Target acquisition times for different numbers of targets and target positions. Note different y axis scales.

so it is possible that with everyday use a user might reach equivalent proficiency in both gestures.

Hypothesis 2 stated that there would be no significant difference between perceived workload in the Bend and Twist conditions. However, the NASA TLX results showed that workload was significantly higher for the Twist condition and, thus, Hypothesis 2 can be rejected. This indicates that participants were not only slower to perform twisting but also perceived that twisting is more demanding.

Hypothesis 3 stated that the target acquisition time would increase as the number of targets increased, and this was borne out by our results. It seems unsurprising that performance improves as target width increases, especially as the task required that the user hold the cursor within the target for two seconds. The larger target means that jitter once a user has reached a target is less of a problem.

Hypothesis 4 stated that target acquisition time would increase as the distance to target increased. Our data shows that, surprisingly, this is not the case: acquisition time does not increase as distance to target increases, and for the case of the closest-left target, it actually significantly decreases for further away targets. Thus, Hypothesis 4 is rejected.

Why are further targets easier?

To explore this surprising result, we looked further at a breakdown of target acquisition time into several stages: reaction time before any movement occurred, movement time until the target was first entered, and jitter time during which the target was left and re-entered any number of times before the final entry (when the force level was held in-target for 2 seconds continuously). Due to lack of space we are unable to show the graphs.

Reaction times were generally low (0.6s for bend, 0.8s for twist) and consistent (they do not change based on size or position of target). The exceptions to this are for the lowest-force targets for which reaction times are slightly higher, which can be explained by the user attempting to apply a very small force, and therefore applying an undetectably small force to start with. This reinforces the conclusion that small forces are harder to apply than large forces.

Movement time to first entering a target generally shows a minor increase as the distance to the target increases. The exception is for the far-left case in many-target cases of the Twist condition, for which the average time spikes higher. By observing participants we could see that this was due to participants sometimes being unable to reach these targets despite applying hard force, and later finding that they needed to be more precise about the angle at which the force was applied. Again, this may be an eliminatable effect due to the peculiarities of our prototype, and/or may be something that users could improve through further experience with twisting.

Reaction time plus movement time typically accounts for up to 1 second of acquisition time for bend targets, and up to 1.5 seconds for twist targets. Therefore, by examining Figure 9 we can see that jitter time accounts for much of

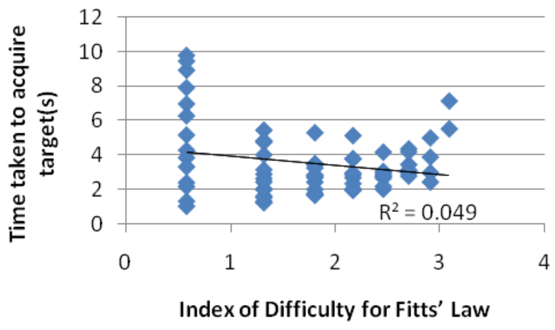


Figure 10: Fitts' Law analysis for Twist force

the acquisition time with 5 targets or more. Unlike other input systems such as a mouse which requires zero effort to hold in one place, the user must apply active effort to hold the force in one place for 2 seconds, and this is the cause of jitter.

We can speculate as to a number of sources of the high jitter time when users apply low forces. The prototype device experiences some elastic deformation which interferes with sensing at small force levels (as it effectively absorbs some of the force making it harder for the user to be accurate with small forces). Some degree of elasticity exists in all devices, depending on the material, thickness and design of their casings, and this effect may prove to be more general than simply a defect in our prototype. Another explanation is that, given the users are already applying a force with their hands to support the weight of the UMPC, it may be difficult for users to modulate the forces they apply by small amounts while keeping the UMPC balanced in their hands, and it is easier to apply larger forces of similar or greater magnitude than the supporting forces.

Fitts' Law

The Fitts' Law model (Shannon formulation, [7]) is expressed as follows, where ID is the index of difficulty, A is the distance to the target and W is the target width.

$$ID = log_2\left(\frac{A}{W} + 1\right)$$

Figure 10 shows the average time for each target plotted against the index of difficulty for the Twist condition (the Bend condition is similar). This illustrates that the data do not fit Fitts' Law: the R^2 value shows that the data is a poor fit, and the average time does not increase with ID as Fitts' Law predicts. This can be further seen because different targets with the same index of difficulty do not share a similar average acquisition time.

Our results show that acquisition time increases as target width decreases, but that it does not increase with distance (and in fact decreases instead). Therefore, we can conclude that the distance factor is the reason that our data are not a good fit to Fitts' Law. Fitts' model was designed to model human movements, in particular for mouse type movements. Mackenzie noted that force sensing devices (e.g. isometric joysticks) undergo negligible limb motion compared to a device like a mouse, and that Fitts' Law may be a poor fit for modelling the performance of such devices.

Comparison with a Mouse

To set some baseline by which our target acquisition numbers can be compared, we refer to a report from Card, Moran and Newell [1] that the average time to point with a mouse to a target on a display is 1.1 seconds. This varies with distance and target size according to Fitts' Law within a range from 0.8 to 1.5 seconds. With force sensing, comparable acquisition times can be achieved using 2-4 targets each side for bending, and 2 targets each side for twisting. Although force sensing offers significantly less resolution than the mouse, if these times are regarded as acceptable interaction latencies, then this indicates that force sensing can be deployed for device interaction successfully.

Implications for design of force sensing interfaces

A number of implications for the design of interfaces using force sensing can be drawn out of these results. The bend gesture performs significantly better than the twist gesture and, therefore, can be used when more targets are required. Target acquisition time increases as the number of targets increases, so there is a tradeoff between speed and number of targets. Since acquisition time increases for closer targets, common operations in a user interface should use targets that are further from the center point.

To avoid jitter time overhead in force sensing based systems, we could explore adding a "select" mechanism which might be a button, another force gesture (e.g. a squeezing gesture), or something else. Alternatively, we can (as for the example interactions for application switching and page turning) make the interactions threshold-based, therefore eliminating any jitter as the force can immediately go back to zero after the threshold is reached.

Finally, we recall that one of the study participants experienced significantly worse response times. This indicates that force sensing may not be comfortable for all users, so alternatives should be provided (as for any input type).

CONCLUSIONS

Force sensing can be used for user inputs through applying physical forces such as twist and bend to mobile devices. Our prototype implementation detects these forces and allows the user to use them to perform application switching (alt-tab) and page turning (page down/up). These interactions benefit from visual feedback which is related to the physical forces the user applies, therefore making them easy to learn and use. We presented a user study into human abilities to apply bending and twisting forces at certain levels. We found a surprising result that low levels of force are harder for users to apply than higher levels of force, and we present some recommendations for the design of force sensing-based user interfaces.

This work opens up a rich new area for research into force sensing as an interaction mechanism. Future work can explore other force sensors and how best to integrate force sensing into mass-produced mobile devices. In addition, interesting interaction design problems can be explored such as what interaction types will benefit from force sensing, and how best to integrate visual, audio and haptic feedback for force-based interaction.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge Gavin Smyth, Steve Hodges, Nic Villar, Shahram Izadi, David Kirk and AJ Brush for assistance during the course of this work.

REFERENCES

1. Card S.K., Moran T.P. and Newell A. The Psychology of Human-Computer Interaction, Lawrence Erlbaum Associates, 1986.
2. Fitts, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, *47*, 381-391.
3. Harrison, B. L., Fishkin, K. P., Gujar, A., Mochon, C., Want, R. Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. In *Proceedings of CHI 1998*, ACM.
4. Hart, S. G., and Staveland, L. E. Development of a multi-dimensional workload rating scale: Results of empirical and theoretical research. In Human mental workload, pp 139-183. Elsevier.
5. Hinckley, K., Pierce, J., Sinclair, M., Horvitz, E., Sensing Techniques for Mobile Interaction, In *Proceedings of UIST 2000*, ACM.
6. Jeong, D. H., Jeon, Y. H., Kim, J. K., Sim, S., Song, C. G., Force-based Velocity Control Technique in Immersive V. E. In *Proceedings of GRAPHITE 2004*, ACM.
7. MacKenzie, I. S. Movement time prediction in human-computer interfaces. In *Readings in human-computer interaction* (2nd ed.), Kaufmann.
8. MacLean, K. E., Shaver, M. J., and Pai, D. K. Handheld Haptics: A USB Media Controller with Force Sensing. In *Proceedings of HAPTICS 2002, IEEE.*
9. Mizobuchi, S., Terasaki, S., Keski-Jaskari, T., Nousiainen, J., Ryynanen, M., Silfverberg, M. Making an impression: force-controlled pen input for handheld devices. In *CHI Extended Abstracts 2005*, ACM.
10. Patel, S. N. and Abowd, G. D. BLUI: Low-cost Localized Blowable User Interfaces. In *Proceedings of UIST 2007*, ACM.
11. Rekimoto, J. Tilting operations for small screen interfaces. In *Proceedings of UIST 1996*, ACM.
12. Schwesig, C., Poupyrev, I., and Mori, E. 2004. Gummi: a bendable computer. In *Proceedings of CHI 2004*, ACM.
13. Snibbe, S. et al. Haptic Techniques for Media Control. In *Proceedings of UIST 2001*, ACM.
14. Williamson, J., Murray-Smith, R., Hughes, S.: Shoogle: excitatory multimodal interaction on mobile devices. In *Proceedings of CHI 2007*, ACM.