Mobile Device Interaction with Force Sensing

James Scott, Lorna M. Brown[*], and Mike Molloy

Microsoft Research, Cambridge, UK

Abstract. We propose a new type of input for mobile devices by sensing forces applied by users to device casings. 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 describe examples of interactions using these forces, employing twisting to perform application switching (alt-tab) and interpreting bending as page-down/up. We present a user study exploring users' abilities to reliably apply twisting and bending forces to various degrees, and draw implications from this study for future force-based interfaces.

Keywords: Force, sensors, mobile devices, interaction.

1 Introduction

Many pervasive computing applications rely on mobile devices. To give three examples, such devices can be used in the control of intelligent environments, they can be used to locate users and objects and provide location-aware applications, and they can be used for communications to support applications requiring pervasive connectivity. Thus, improving users' experience of interacting with mobile devices is a key ingredient in pervasive computing.

For mobile devices such as tablet or slate PCs, ultra-mobile PCs (UMPCs), PDAs, or new smart phones such as the 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 better 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, buttons, jog dials, etc. While these devices typically have touch-sensitive screens, dedicating a portion of the screen to an input interface reduces the available screen area for information presentation.

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

[*] Now at Vodafone.

In this paper, we describe the concept of force sensing as a user input. We detail our prototype implementation based on a UMPC using thin load sensors. We present examples of device interactions where force sensing may prove useful, namely to provide functionality normally associated with shortcut keys such as alt-tab and page-down/up on mobile devices without keyboards. We then describe our user study into the capability of users to control the forces applied to devices, allowing us to evaluate the usefulness of force sensing as an input mechanism.

2 Related Work

In the past decade there has been a move to consider new interaction techniques for mobile devices beyond the use of physical or virtual on-off buttons, including work using accelerometers, e.g. Rekimoto [1] and Hinckley et al. [2], or inertial sensing [3].

The Gummi concept and prototype [4] propose 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, allowing for simpler integration with current device designs which incorporate many rigid components.

Previous work on force sensing for rigid devices uses direct pressure (see Figure 1, top left), whereas our work explores several different types of forces such as bending and twisting (Figure 1, bottom left and right). Researchers at Xerox PARC used pressure sensors [5] both as a physical "scroll bar" so that squeezing 2/3 of the way along the top edge would navigate 2/3 of the way along a document, and to detect which side a user was gripping a device to inform "handedness-aware applications". Direct pressure input has also been explored in the context of a touchpad [6] or touch screen [7]. Jeong et al. [8] also used direct pressure applied to a force sensor mounted on a 3D mouse to control movement speed in virtual environments.

A number of input devices have been proposed which use force sensing in various ways but do not have any computing or display electronics. The Haptic media controller by Maclean et al. [9] 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. The haptic rotary knob by Snibbe et al. [10], which can sense the rotational forces applied by a user. TWEND [11] and Bookisheet [12] are flexible input peripherals which can sense the way they are bent.

Another related field of work is that of Tangible User Interfaces [13], which if defined widely can include any type of physical UI such as force sensing. In Fishkin's terminology [14], the current focus of our work is on "fully embodied" interaction in that the output is shown on the device itself, in contrast to the input-only systems above. However, our ideas could in future also be applied to augment passive objects while retaining their rigidity, e.g. to provide computer inputs for the control of intelligent environments.

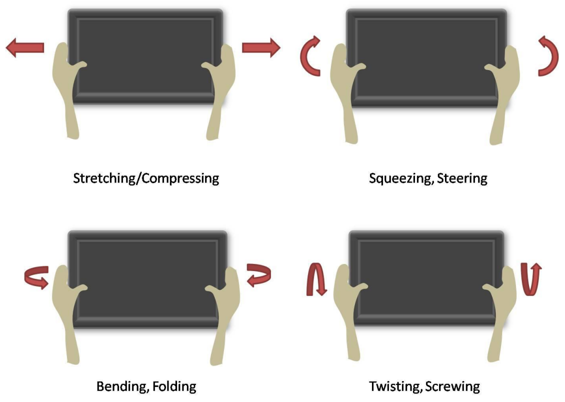


Fig. 1. Four force gestures that can be sensed

3 Sensing User-Applied Forces

Physical forces that users apply to device casings 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. 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.

Users can apply various types of force, with four examples shown in Figure 1. Each force can be applied in two directions (the way indicated by the arrows and the opposite directions), and can be applied to a variable degree. Forces can be applied along different axes (e.g. vertically as well as horizontally) 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. In this work, we focus on two of the forces shown, namely bending and twisting.

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.

3.1 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 can facilitate more robust, more easily manufactured, and smaller 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. When physical controls are present on a device, their positions lead to an implied orientation and grip for the device, while with force sensing, the device can be easily made more symmetric, allowing a device to avoid having an intrinsic "right way up" which may be useful in some 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, unobscured by a finger or stylus. During force sensing input the device can be essentially stationary, thus inputs can be made subtly which may be useful in some situations (e.g. during a meeting).

One potential disadvantage of force sensing is its ability to be triggered accidentally, e.g. while the device is being carried or handled. This is in common with many other input mechanisms for mobile devices, such as accelerometer-based, touch-based or even button-based. With force sensing, it may be necessary to use a "hold" button or "keylock" key-combination for users to indicate that input (including force sensing input) should be ignored.

Another disadvantage of force sensing may be in the need for using two hands, which is the mode of operation of our prototype based on a UMPC (see below). One can envision one-handed use of force sensing, e.g. using another object/surface as an anchor, or using a device small enough to both support and apply forces to in a single hand. However, the desirability and usability of this for input remain questions for future work; advantages such as the ability to keep the screen at the correct viewing angle and unobscured may be lost.

A further issue with force sensing is in the effects of repeated shearing and bending forces that the user applies to the device over time. While this may result in higher mechanical demands on the casing of the device, there is also an advantage to the device being "aware" of its physical environment, so that it can warn the user (e.g. audibly) if excess forces are applied, either during force-based user interface actions or otherwise (e.g. placing heavy books on the device).

Although this section compares force sensing to other forms of input, it is important to note that an either-or decision is not implied, 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 on a device rather than a way of replacing another input.

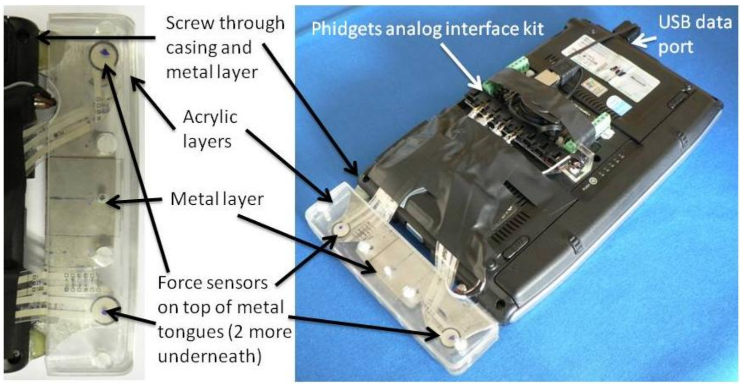

Screw through casing and metal layer
Acrylic layers
Metal layer
Force sensors on top of metal tongues (2 more underneath)
Phidgets analog interface kit
USB data port

Fig. 2. Force sensing prototype using augmented UMPC

4 Prototype Force Sensing Hardware

We built a prototype force sensing device by augmenting a Samsung Q1 UMPC with a custom additional casing incorporating four force sensors, as shown in Figure 2. We used an additional casing rather than an integrated solution for feasibility reasons; UMPCs are tightly packed with components and have precisely fitted cases, making it difficult to incorporate force sensing hardware internally. However, in an device with force sensing designed in from the beginning, an additional casing would be unnecessary. We pilot-tested both strain gauges (which detect tension forces) and pressure sensors (which detect compression forces) and found the latter to be easier to obtain useful data from and easier to integrate in the mechanical construction of a first prototype.

The additional casing comprises an exterior acrylic section made of layers cut with a laser cutter, screwed together and hand-filed to round the corners. This tightly surrounds a central hand-cut metal layer (magnesium alloy) which is rigidly attached to the body of the UMPC at the same screw points that hold the top and bottom half of the UMPC casing together.

For the force sensors we use four FlexiForce 0-25 lb load sensors from TekScan Inc. These are small (the sensing area is 10 mm wide and can be made smaller) and thin (0.2 mm) 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 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 casing are dictated by the forces that we intended to sense; this prototype was built to sense "twist" and "bend" gestures (see Figure 2).

The Flexiforce sensors lower their electrical resistance from over 30 MΩ when no pressure is applied to around 200 kΩ 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 1 MΩ pull-down resistor. These voltages are measured in the prototype using analog to digital converters in a Phidgets

Analog Interface Kit which is attached to the back of the UMPC (as shown in Figure 2) and software on the UMPC queries the force values through a USB interface.

4.1 Determining Force Gestures Using Sensors

With load 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 this to be unreliable since friction causes the "zero" point to move each time the device is manipulated. Therefore, the prototype design senses 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:

```
TwistClockwise = BottomBack + TopFront
TwistAnticlockwise = BottomFront + TopBack
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 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, to best detect the bending gestures, while the bottom tongue's two sensors were placed at the top and bottom edges of the tongue, to best detect twist gestures. Thus, the second term in each of the equations above was removed. However, while this greatly improved the true positive rate, it did not eliminate false positives, i.e. the sensors also sometimes trigger when the wrong gesture is applied, e.g. the `BottomFront` sensor indicating `TwistAnticlockwise` would sometimes trigger for `BendAway`. We therefore further modified the rules, such that when two sensors trigger, the gesture detected corresponds to the one compatible with these two sensors, and the other gesture is "damped" by subtraction. This results in reliable performance in practise.

```
TwistClockwise = BottomBack - TopBack
TwistAnticlockwise = BottomFront - TopFront
BendTowards = TopBack - BottomFront
BendAway = TopFront - BottomBack
```

4.2 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. To illustrate the potential

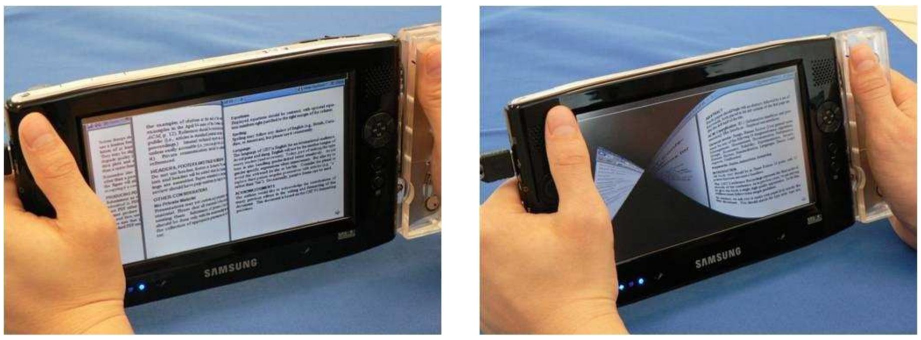


Fig. 3. Two examples of force-based interactions: (left) bending for page turning and (right) twisting for application-switching, with visual feedback

usefulness of force sensing, we built demo software showing how it can be used for one class of interactions, that of replacing "shortcut keys".

Our motivation for considering this example compelling is as follows. Mobile devices such as UMPCs are capable of running applications that desktop PCs run, but have reduced I/O capabilities. One of the implications of this is that the convenient shortcut keys that users may normally employ with a full keyboard may be unusable on a UMPC since they keys are not present or difficult to use quickly due to their size or placement. Force-based interactions are an interesting alternative, taking advantage of the fact that UMPCs are designed to be gripped by the hands during use.

Page-Down and Alt-Tab Force-Based Interactions. We implemented two force-based shortcut key interactions on the UMPC, as shown in Figure 3. 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. For page-down, we provide visual feedback as if the user was 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. This is so a user can go forwards or backwards from the window they are on in a consistent fashion. Users receive differing visual feedback with the twist direction matching the direction of the force applied.

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 and the animation to be run with 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. 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. This allows users to apply small forces to see the potential effects (e.g. seeing which window would appear when twisting that way) before committing with a larger force.

The UMPCs were used for a 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.

Other Interactions. We do not claim that the mappings chosen are in any way “best”, merely that they illustrate the potential of force sensing as an input. Other mappings could, of course, be used instead, e.g. twisting could be used to indicate “cut” and bending “paste”.

We did choose the mappings based on the ease of providing visual feedback that has strong physical analogies with the applied forces, as we expected (and found during demos) that this made force-based interaction intuitive to discover through observation, and easy to remember and to use. Such physical analogies are popular choices in past work, e.g. Harrison’s page turning gesture [5]. This analogous feedback could be extended to other interaction mappings, e.g. twist for “cut” could involve the cut text twisting in on itself, and bend for “paste” could involve the the document visually bending and splitting to make space for the pasted text to appear at the cursor. Again, we expect that the visual mapping will assist users in remembering which force performs which UI action.

Another modification to be explored 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.

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. Richer audio cues can be incorporated in future, e.g. cues which also match the physical action taken such as a crumpling sound or page-flicking sound. Haptic feedback can also be used, particularly since the user is known to be gripping the device, furthering the analogy with physical movements.

While these are all exciting possibilities, before looking further into them, in this paper we wish to address more basic questions: how repeatably and accurately can a number of users apply forces such as bending and twisting to devices. A study of these issues is the subject of the rest of this paper.

5 User Study

We conducted a quantitative user study to assess the capabilities of force sensing as an input method using our prototype. We chose this form of study so as to gain understanding of the fundamental capabilities of users to apply forces in a controlled fashion to mobile devices, thus informing the design of force sensing interfaces. The study aimed to assess the number of distinct levels of force that a user could reliably "hit", and the speed at which they could do this, for both twisting and bending forces. The methodology was based on that used by Ramos et al. [15]. A screenshot of the software used in the study is shown in Figure 4.

5.1 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, human resources and marketing. Participants were each given a box of chocolates worth around 5 GBP in thanks for their time.

While conducting the tests, participants were provided with an office chair with adjustable arms and a desk and could choose to sit in any comfortable

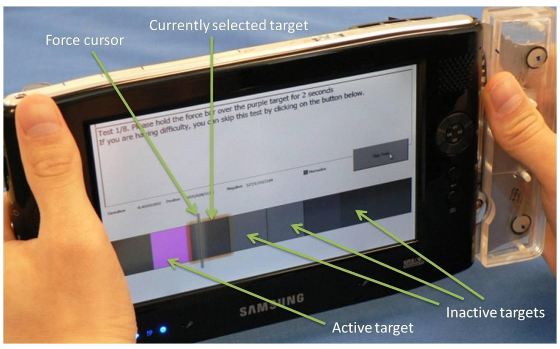


Fig. 4. Screenshot of software for user trials

way. Some participants leaned on the desk, some sat back in the chair, and some changed positions during the trial.

5.2 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 to allow for differences in individual strength.

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. The mapping from twisting/bending to left/right was decided by observing the most natural mapping in pilot tests; this was not a source of confusion to the participants after the training phase.

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 [16]. 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 4 illustrates a test with four targets on each side (i.e. targets with the same width but varying "amplitudes" in the normal Fitts' Law terminology). 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 coloured purple and the user was asked to move the force cursor as quickly as possible into the coloured 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 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 deliberately chose not to include a button for users to click on a target as soon as they enter it, for three main reasons. First, because the trial was conducted using a two-handed grip, adding a button would add a significant new demand on one of the hands applying forces. Second, with force sensing users can change the position and grip style of their hands, so it was not clear where a button would be placed. Third, we wanted to explore the potential for button-free interaction with force sensing.

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 2, 4, 6 and then 8 targets in each direction, allowing the user to become accustomed to the system through progressively harder tasks. After the training phase, the user was given the opportunity to recalibrate the force maximums.

During the main experiment phase, the number of targets in a given block was varied between 2, 3, 4, 5, 6, 7 and 8 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, covering a wide range of widths and amplitudes. The total duration of the study was around 50 minutes per participant.

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.

6 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 worse than the mean for the majority of 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.

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

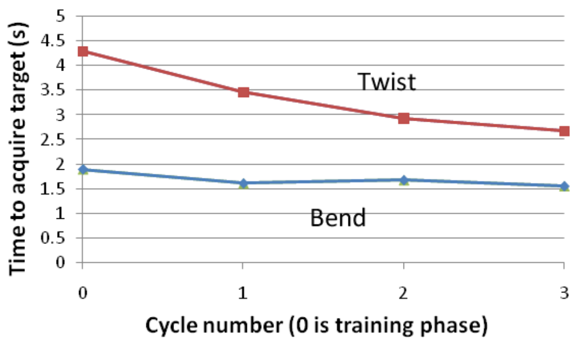


Fig. 5. Effect of participant experience on target acquisition time

6.1 Learning Effects and Effects of Gesture Type

Figure 5 shows the effect of the cycle on the average target acquisition time (cycle 0 is the training phase). Two within groups ANOVAs (analyses of variance) were carried out, one for each gesture type, to investigate whether there was any significant learning effect. For the Bend gesture, the ANOVA showed no significant difference between cycles. The ANOVA for the twist gesture, on the other hand, showed a significant effect of cycle (F(3,24)=7.09, $p<0.01$). Post-hoc Tukey tests showed that there was a significant difference ($p<0.01$) between the training cycle and cycles 2 and 3.

Excluding the training phase, 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(1,16)=14.34, $p<0.01$).

6.2 Effect of Number of Targets/Target Width

Figure 6 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). Two within-group ANOVAs were carried out, one for each gesture. These showed significant effects of number of targets on performance for both gestures (for Bend, F(6,160) = 55.11, $p<0.01$ and for Twist, F(6,160) = 23.06, $p<0.01$). Post hoc Tukey tests showed significant ($p<0.05$) differences between numbers of targets as follows.

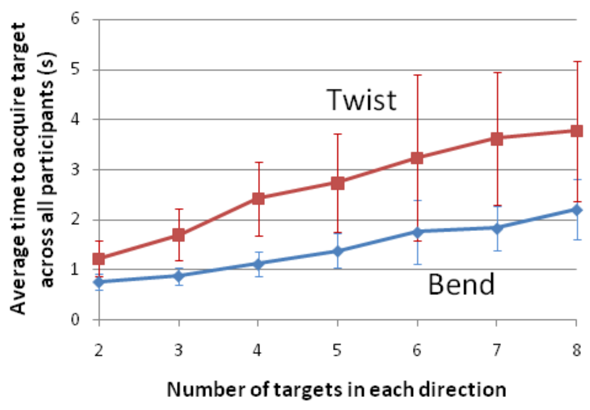


Fig. 6. Effect of different numbers of targets on average target acquisition time

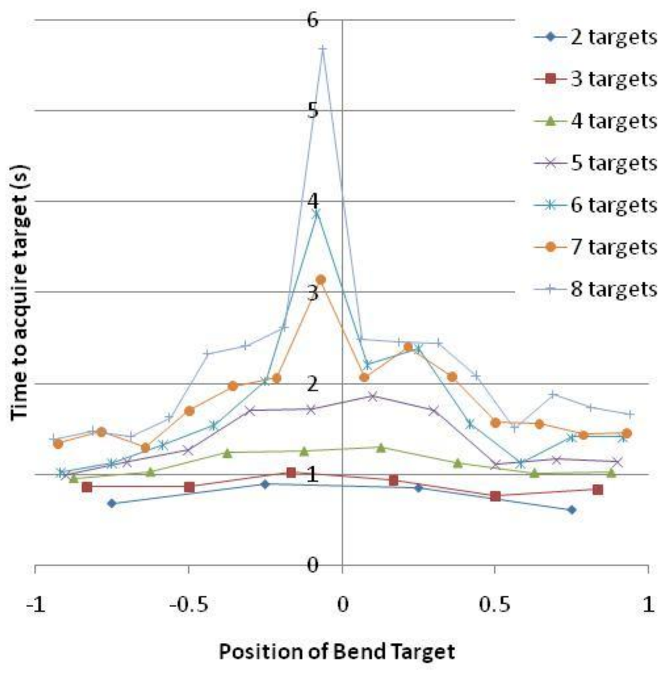
For Bend:
2 was significantly faster than 4, 5, 6, 7 and 8
3 was significantly faster than 5, 6, 7 and 8
4 and 5 were significantly faster than 6, 7 and 8
6 and 7 were significantly faster than 8

For Twist:
2 was significantly faster than 4, 5, 6, 7 and 8
3 was significantly faster than 5, 6, 7 and 8
4 and 5 were significantly faster than 7 and 8

6.3 Effects of Target Position

Figure 7 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. Within-groups 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



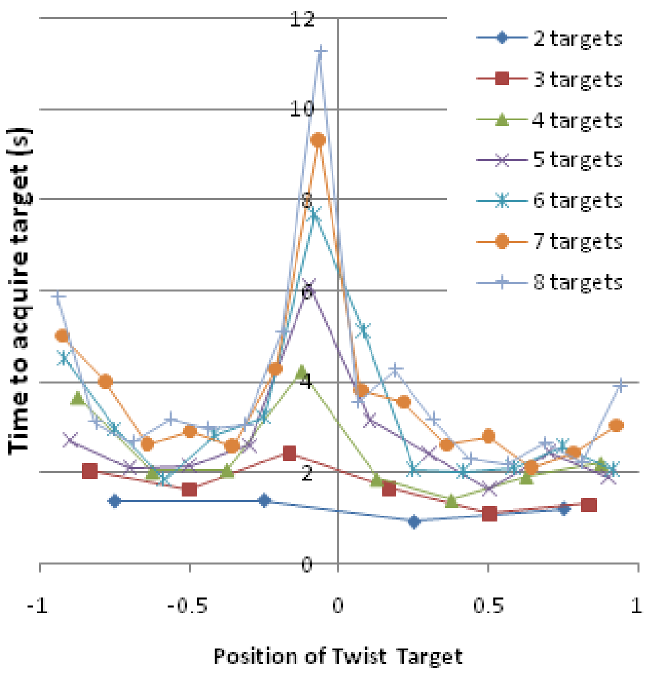


Fig. 7. Target acquisition times for different numbers of targets and target positions. Note different y axis scales.

than almost all other targets. It should be noted that “left” for twist and bend are two very different gestures.

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.

7 Discussion

Our data shows that target acquisition is significantly faster in the Bend condition than in the Twist condition. 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, thus making the “far” targets simpler to reach. Twisting performance improved significantly during the experiment, so it is possible that with everyday use a user might reach equivalent proficiency in both gestures.

7.1 Fitts’ Law

As Fitts’ Law is applicable to many input devices and its implications affect the design of user interfaces, we tested its applicability for force sensing. The Fitts’ Law model (Shannon formulation [16]) is expressed as follows, where ID is the index of difficulty, A is the amplitude or distance to the target and W is the target width (inversely proportional to the number of targets).

$$ID = log_2(A/W + 1)$$

If Fitts’ Law were to apply to force-based interfaces, we would expect that the target acquisition time would increase as the number of targets increased, and this was borne out by our results. However, Fitts’ Law would also cause us to expect that target acquisition time would increase as the distance to target increased. Our data shows that this is not the case: acquisition time does not increase as distance to target increases, and for the case of the closest-left target for both gestures, it actually significantly decreases for further away targets. MacKenzie [16] noted that force-based 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 modeling the performance of such devices; our results confirm that this is true for twisting and bending forces.

7.2 Reaction Time, Movement Time and Jitter

To further understand our results, we looked 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). 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. This may be eliminated with practise by the user or with a further refined implementation which is more forgiving with regards to the angle that the force is applied.

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 7 we can see that jitter time accounts for much of 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 errors in keeping a constant force cause jitter.

We can speculate as to a number of sources of the high jitter time when users apply low forces for both gestures. The prototype device experiences some elastic deformation which interferes with sensing at small force levels, however, the fact that the same result is found for two different force gestures suggests that it is not a problem with the particular sensor mounting used. Some degree of elasticity is present in all devices. 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.

7.3 Implications for Design of Force-Based 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 targets with low force, user interfaces should use avoid the use of such targets, or use wider targets at low forces.

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, as with our example use of force sensing for alt-tab and page-down/up, we can make the interactions threshold-based, therefore eliminating any jitter as the force can immediately go back to zero after the threshold is reached.

8 Conclusions

Force sensing can be used for user inputs through applying physical forces such as twist and bend to mobile devices. We described a prototype implementation using pressure sensors added to a UMPC, and example uses of force sensing to perform shortcut key functionality that is otherwise missing from mobile PCs, for 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 easier to learn and use.

We presented a user study into human abilities to apply bending and twisting forces at certain levels. We found that users performed bending quicker than twisting, that up to 4-5 separate levels of force were applyable by users without excess "jitter" in holding the force at that level, and that low levels of force were more difficult for users to apply than higher levels of force.

This work opens up a rich set of further research into force sensing as an interaction mechanism. One area of exploration is how best to map force inputs to user interface actions, for shortcut key replacement or otherwise. Alongside this it will be useful to investigate how best to integrate visual, audio and haptic feedback for force-based interaction. Future work can also explore other types of force sensor and how best to integrate force sensing into mass-produced mobile devices. Finally, it will be interesting to see how force sensing can be used in combination with other input types (e.g. touch-based or position/orientation-based) in order to take best advantage of each mechanism.

References

1. Rekimoto, J.: Tilting operations for small screen interfaces. In: Proceedings of UIST 1996. ACM Press, New York (1996)
2. Hinckley, K., Pierce, J., Sinclair, M., Horvitz, E.: Sensing techniques for mobile interaction. In: Proceedings of UIST 2000 (2000)
3. Williamson, J., Murray-Smith, R., Hughes, S.: Shoogle: Excitatory multimodal interaction on mobile devices. In: Proceedings of CHI 2007. ACM Press, New York (2007)
4. Schwesig, C., Poupyrev, I., Mori, E.: Gummi: a bendable computer. In: Proceedings of CHI 2004. ACM Press, New York (2004)

5. 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 Press, New York (1998)
6. Rekimoto, J., Schwesig, C.: Presenseii: bi-directional touch and pressure sensing interactions with tactile feedback. In: Extended Abstracts of CHI 2006. ACM Press, New York (2006)
7. Mizobuchi, S., Terasaki, S., Keski-Jaskari, T., Nousiainen, J., Ryynanen, M., Silfverberg, M.: Making an impression: force-controlled pen input for handheld devices. In: Extended Abstracts of CHI 2005. ACM Press, New York (2005)
8. 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 Press, New York (2004)
9. MacLean, K.E., Shaver, M.J., Pai, D.K.: Handheld haptics: A usb media controller with force sensing. In: Proceedings of HAPTICS 2002. IEEE, Los Alamitos (2002)
10. Snibbe, S.S., Shaw, R., Roderick, J.: Haptic techniques for media control. In: Proceedings of UIST 2001. ACM Press, New York (2001)
11. Herkenrath, G., Karrer, T., Borchers, J.: Twend: Twisting and bending as new interaction gesture in mobile devices. In: Extended Abstracts of CHI 2008. ACM Press, New York (2008)
12. Watanabe, J., Mochizuki, A., Horry, Y.: Bookisheet: Bendable device for browsing content using the metaphor of leafing through the pages. In: Proceedings of UbiComp 2008 (2008)
13. Ishii, H., Ullmer, B.: Tangible bits: Towards seamless interfaces between people, bits and atoms. In: Proceedings of CHI 1997. ACM Press, New York (1997)
14. Fishkin, K.: A taxonomy for and analysis of tangible interfaces. Personal and Ubiquitous Computing 8(5), 347–358 (2004)
15. Ramos, G., Boulos, M., Balakrishnan, R.: Pressure widgets. In: Proceedings of CHI 2004. ACM Press, New York (2004)
16. MacKenzie, I.S.: Movement time prediction in human-computer interfaces. In: Readings in human-computer interaction, 2nd edn. Kaufmann, San Francisco