Touch Scrolling Transfer Functions

Philip Quinn, Sylvain Malacria, Andy Cockburn

Touch Scrolling Transfer Functions

Philip Quinn **Sylvain Malacria** **Andy Cockburn**

University of Canterbury
Christchurch, New Zealand

philip.quinn@canterbury.ac.nz,
sylvain@malacria.fr, andy@cosc.canterbury.ac.nz

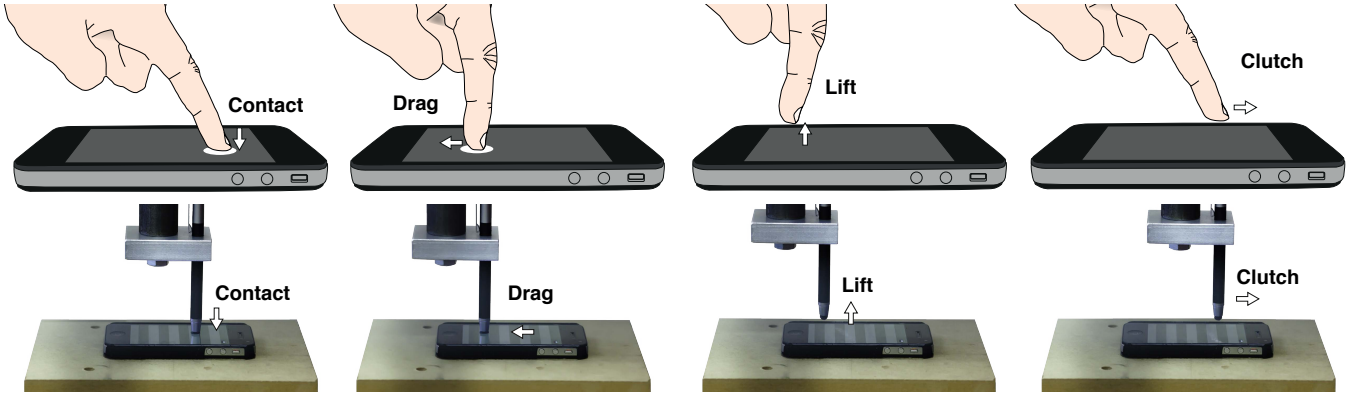


Figure 1. Transfer functions transform touch scrolling gestures (top) into scrolling output. These gestures can be simulated mechanically by SCARA robots (bottom) to reverse engineer the transfer functions.

ABSTRACT

Touch scrolling systems use a *transfer function* to transform gestures on a touch sensitive surface into scrolling output. The design of these transfer functions is complex as they must facilitate precise direct manipulation of the underlying content as well as rapid scrolling through large datasets. However, researchers' ability to refine them is impaired by: (1) limited understanding of how users express scrolling intentions through touch gestures; (2) lack of knowledge on proprietary transfer functions, causing researchers to evaluate techniques that may misrepresent the state of the art; and (3) a lack of tools for examining existing transfer functions. To address these limitations, we examine how users express scrolling intentions in a human factors experiment; we describe methods to reverse engineer existing 'black box' transfer functions, including use of an accurate robotic arm; and we use the methods to expose the functions of Apple iOS and Google Android, releasing data tables and software to assist replication. We discuss how this new understanding can improve experimental rigour and assist iterative improvement of touch scrolling.

Authors' version.
UIST'13, October 8–11, 2013, St Andrews, Scotland.

http://dx.doi.org/10.1145/2501988.2501995

Author Keywords

Control-display gain; flick scrolling; scroll acceleration; touch interaction; transfer functions.

ACM Classification

H.5.2 [**Information interfaces and presentation**] User interfaces – *Interaction styles*.

INTRODUCTION

Gesture-based 'touch scrolling' allows users to scroll by simulating a direct physical interaction on the displayed content. Metaphors from the real world—such as the ability to *throw* the content with a velocity and *friction* that gradually slows it—assist with giving the user a scrolling mechanism that is perceived to be 'simple and natural'. Yet despite the perceived simplicity of gesture-based scrolling, sophisticated engineering underlies its behaviour. Close visual inspection of flick scrolling on an Apple iPhone, for example, will reveal that the scrolling velocity following a 'flick' gesture does not always follow simple physical laws, and that accelerated scroll speeds and rapid deceleration are sometimes applied.

The behaviour of touch scrolling techniques is determined by a *transfer function* [8, 17] that converts human input actions into a resultant scrolling output effect. These functions may attend to input parameters such as properties of individual gestures (such as their length, velocity, pressure, or direction), properties of a gesture sequence (such as the number of gestures issued in a rapid sequence or their period), properties of the document and viewport (such as the document's length or

type of content), as well as attending to current system state and settings (such as scroll speed or friction setting). While any or all of these parameters *may* be used in any particular implementation, there is a lack of public information on which ones are actually used and on how they are used. Furthermore, there is a lack of fundamental human factors knowledge on how users choose to express their intentions for different scrolling velocities through gestures.

Given its role as a core component of touchscreen interaction, the transfer function that governs touch scrolling behaviour has received surprisingly little research attention (exceptions reviewed later). One reason for this is that the proprietary systems representing the state of the art are black-boxes that do not expose key aspects of their operation. Most platforms publicise and allow control over only a limited set of their built-in scrolling parameters. Another key reason is that there is a lack of methods that might allow inspection of the transfer functions. While recent work has disclosed techniques for inspecting transfer functions in mouse-pointing [4] and scroll-wheel operation [17], these methods rely on published USB standards for communication with external devices, and they therefore cannot be applied to touch scrolling.

The lack of information on touch scrolling transfer functions is a problem for researchers. It is a source of experimental imprecision because exact system settings can not be reported or replicated, and it impedes the iterative refinement of touch scrolling systems because key behaviours (both human and system) are unknown. For example, Aliakseyeu et al. [1], Lee et al. [13], and Tu et al. [19] all implement and evaluate touch scrolling transfer functions, where the baseline experimental control condition unintentionally differs from the behaviour of commercial devices.

We present four main contributions to resolve these problems. First, we present a human factors experiment to understand critical characteristics and limits of user actions when performing touch scrolling gestures. Second, we describe methods that facilitate reverse engineering the parameters used by touch scrolling transfer functions. These include controlled software simulation, and use of a high-quality Selective Compliant Articulated Robot Arm (SCARA) to precisely control input gestural actions. Third, we present results of using the methods to compare the transfer functions used by Apple iOS (iPhone and iPad) and Google Android. The results show some similarities, but also substantial differences in their transfer functions, suggesting that any experiment involving gestural scrolling is likely to be influenced by the platform on which it is conducted. Fourth, we release the source code for the tools used to extract, emulate, and test these transfer functions; to serve as documentation of the current state of the art and a foundation for iterative improvement.

BACKGROUND

The term 'scrolling' is often used informally in HCI to refer to any geometrical translation of the information displayed within a viewport, which permits diagonal or elliptical movement where the information is translated on two dimensions concurrently. However, this usage is inconsistent with the origin of the word 'scroll', which constrains movement to one dimension at a time. In this paper we use the word 'scroll' to refer to one-dimensional translation of the viewport. This section provides background on scrolling transfer functions and prior techniques.

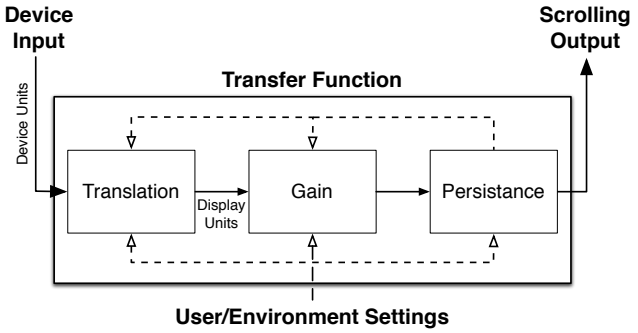


Figure 2. Conceptual low-level depiction of scrolling transfer functions (adapted from Quinn et al. [17]).

Scrolling Transfer Functions

Transfer functions are responsible for converting human device manipulations into resultant output effects [8]. Several researchers have noted the important role that scrolling transfer functions have on user performance [e.g. 7, 20], but prior scrolling studies have been inexact on the transfer functions used due to a lack of methods for analysing, reporting, and controlling their settings.

Casiez and Roussel [4] recently made similar observations on the lack of tools for analysing and controlling mouse pointing transfer functions, causing in imprecise 'experimental bricolage'. Consequently, Casiez and Roussel [4] developed a USB mouse emulator (the *EchoMouse*) and software libraries to resolve these problems. Quinn et al. [17] adapted the EchoMouse to analyse scrolling transfer functions used with commercial scroll-wheel devices. Figure 2, adapted from Quinn et al. [17], shows three conceptual transformations that can occur in such a function: (1) *translation* converts device units into display units, such as millimetres of finger displacement into pixels of scroll translation; (2) *gain* amplifies or attenuates the input signals; and (3) *persistence* allows state memory to influence future behaviour, such as counting the number of repeated flick actions, or applying effects across time (such as momentum). The transfer function can also be influenced by user settings and by the environment (e.g. the length of the document or display pixel density).

Quinn et al. found substantial differences between the parameters used by different scroll-wheel driver vendors, and between the ways the parameters are used. For example, some apply different levels of gain across scroll direction and cumulatively apply gain across repeated scroll-wheel actions, others attend

only to scroll-wheel velocity, and some attend to the duration of input. The diversity of approaches demonstrated that supporting scrolling through a wheel is surprisingly complex.

Touch Scrolling Techniques

With touch input, users can readily express several parameters in addition to those expressible via. scroll-wheels: many gestural forms are possible—linear [e.g. 1, 13], elliptical [e.g. 2, 18], or oscillatory motion [e.g. 14]; pressure can be varied [e.g. 9, 16]; as can the number of points of contact, and the type of limb contact [e.g. 6].

In this paper, we focus on linear (or near-linear) gestures. We use the term 'drag' to describe the action of moving the finger across the touch surface, independent of system feedback, and we use the term 'flick' to describe rapid finger movement at the point that it disengages from the touch sensitive surface. The basic elements of linear gestures are conceptually simple, as depicted in Figure 1: (1) users make *contact* with the surface; (2) they *drag* their finger(s) on the surface; (3) they *clutch* their finger or fingers (possibly initiating a *flick*), which involves lifting the contact(s) from the surface to regain a convenient space for subsequent contact to repeat the drag; and (4) they *disengage* from the surface when complete, and may (5) *grab* the surface by placing their finger back on the display and holding their position. Despite the apparent simplicity of these actions, they allow many parameters of expression that could be used by a transfer function to influence scrolling output.

Contact and Drag

During *contact* and *drag*, users can control parameters including gesture length, contact time, velocity, acceleration, contact area, orientation of contact area, limb use, pressure, and location. Multiple contacts (e.g., two finger scrolling) and bimanual contacts (e.g., two thumbs 'running' on the surface) could also be used. Example transfer functions using many of these parameters are available in commercial and research systems: for example, the perpendicular distance between the slider axis and finger location governs the control granularity of the playhead in Apple's iOS 'Music' application, and researchers have examined the use of pressure to control zoom [16] and inertial speed [3].

The user's perception of a transfer function's behaviour during *contact* and *drag/flick* is likely to be influenced by whether the interaction is *direct* or *indirect*. With direct interaction the touch surface is coincident with the output display, which affords a perception of physical direct manipulation. However, the expected effect of this manipulation has been debated—whether the user's movement should be interpreted as moving the viewport (moving the content in the opposite direction to the finger's movement), or whether the content moves with the finger [11, 12]—with contemporary implementations favouring the latter. This perception is enhanced through physically inspired behaviours, such as the ability to 'throw' the content with simulated momentum and friction. While these behaviours typically use a 1:1 mapping from contact movement on the surface to output effect, Kwon et al. [12] suggests that users are tolerant of mapping discontinuity, particularly when scrolling large distances.

Indirect interaction, such as trackpad scrolling, involves a separation between the input surface and the displayed content, which diminishes or eliminates the need for a 1:1 mapping. Indeed, multi-finger scrolling on trackpads has typically reversed the directional binding used on 'direct' manipulation devices, with downward dragging actions scrolling towards the end of the document rather than the beginning. The disconnect between input and content can be used to allow the user more precise control over their scrolling actions; for example, Hinckley et al. [9] describe several ways in which motion along a touch sensitive strip can be used to scroll, including automatic scrolling in response to pressure and accelerated scrolling in response to a finger's velocity across the strip.

Clutch and Repeat

Clutching and repetition are related as convenient repetition of linear gestures requires that a clutching action be performed. Clutching involves temporarily disengaging from the touch surface by lifting the finger, then repositioning it in the air to re-engage with the surface near the initial contact (however, certain styles of touch interaction negate the need for an explicit disengagement from the display [e.g. 2, 14, 18]).

Prior work with scroll wheels [10, 17] has used the count of successive clutching actions to increase scrolling gain to facilitate long distance scrolling. Equivalent methods could be used with touch scrolling by cumulatively applying gain across a sequence of repeated gestures. Furthermore, metaphors of inertia and friction can be used to support continued scrolling while the finger is out of contact [15, 21].

STUDY ONE: HUMAN SCROLLING MECHANICS

Designing a touch scrolling transfer function requires understanding the characteristics and limits of human input when gestures are expressed on devices. The lower-bounds of gestural scroll control are relatively clear—a user who slowly drags their finger across the touchscreen surface probably intends to scroll slowly, even if the rate is only a few millimeters per second. However, the mechanisms of rapid scrolling are unclear, with unknown factors including the following: what parameters do users choose to manipulate to signify an intention to scroll quickly; how fast do their fingers move; how long are the gestures; and how rapidly are gestures repeated?

The primary goal of this study is to gather data on the range and style of input that user's provide on touch devices to perform scrolling, answering questions such as those in the previous paragraph. In addition, the results are used to establish the range of values used to robotically probe devices in Study 2.

Prior experience with touch scrolling interfaces and scrolling feedback will influence users' expectations for scrolling behaviour. To reduce the impact of these factors the experiment

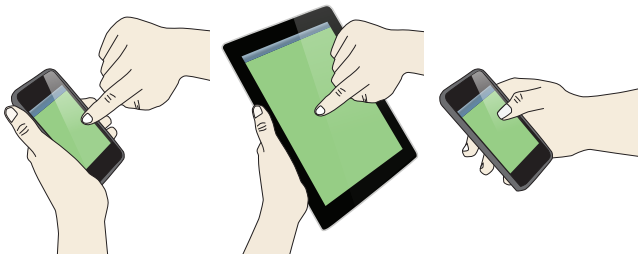


Figure 3. Devices and postures examined: held in one hand and operated with the other (iPhone: left, iPad: centre), and an iPhone held in one hand and operated with the thumb (right)

was conducted without any scrolling feedback. Instead, participants were asked to carry out whatever gestures seemed natural to achieve the requested scrolling velocities without any system response.

Method

We examined the participants' gestural actions when asked to imagine scrolling at three different speeds (browsing speed, comfortably fast, and maximally fast; detailed later) and in two directions (up and down). Three device and posture combinations were analysed: an Apple iPhone 4S held in one hand and operated with the index finger of the other hand; an Apple iPhone 4S operated with the thumb of the hand holding the device; and an Apple iPad 2 held in one hand and operated with the index finger of the other hand (Figure 3). These represent the most commonly observed operating postures of the devices. Devices were always held in a portrait orientation.

Subjects & Apparatus

Twelve volunteers (four female; mean age 26) participated in the experiment. All were right-handed post-graduate computer science students who owned and regularly used a touchscreen device (various vendors). Software logged all gesture events, and video captured the participants' postures and actions.

Quantization effects arising from the devices' digitisation of gestural movement will influence our results, particularly at high movement velocities and with short-duration gestures. To understand the magnitude of this issue we analysed the device event reports. The iPad and iPhone reported movement at 16 ms intervals. We tested the accuracy with which the coordinates of the edges of the display were reported by issuing very fast dragging gestures across the display edges, with results showing edges to be reported with ±1 mm accuracy. However, the coordinates and time of the first event-report were variable when several seconds had passed since the previous contact (we assume the polling interval of the sensor panel is increased to conserve battery power). Similarly, the final event report when breaking contact frequently showed much a lower velocity than prior reports, which could be explained by the event being buffered until the device can confirm that contact has been lost. Our results, therefore, describe the human-factors of gestures *as observed by contemporary devices* (at the application-level).

Procedure

Participants were instructed that the experiment concerned how users perform scrolling gestures at different speeds and directions on touchscreen devices. They were told that they would not see the effect of their scrolling actions, but that they should issue gestures to control three speeds:

- *Browsing Speed:* "Imagine that you are scrolling through a list of songs, looking for something good to play."
- *Comfortably Fast:* "Imagine that you want to quickly scroll to the bottom of a long webpage or list."
- *Maximum Speed:* "Scroll as fast as you can."

One consequence of not showing scrolling feedback is that scrolling direction is ambiguous (pilot studies showed a tendency for participants to move their fingers downwards when instructed to scroll down, which would cause a document to scroll upwards, towards its start). We therefore instructed participants to move their fingers downwards for 'Down' tasks, and on how to hold the device with each posture.

The experimental software showed a title bar with the target speed for the next gesture set, the direction, and a numerical timer on the touchscreen. The rest of the display was a solid yellow colour except for a 'Tap to Begin' button in the centre of the display. Tapping the button caused the display to turn green; the timer began counting down from 6 seconds upon the participant's first contact with the screen. On reaching zero, the display turned red for 1 second before returning to the yellow 'tap to begin' state. Two initial trials at 'Browsing speed' were included to familiarise participants with the procedure (data discarded).

Each participant completed 18 six-second trials with each of the three device/posture combinations. The trials consisted of three repetitions of each direction and speed; all were completed with one posture before moving on to the next (posture order counterbalanced across participants). Within each posture, all trials at one speed (both directions, alternated across trials) were completed before advancing to the next speed.

Design

To reiterate, the experimental objective is to characterise human gestural actions in touch scrolling, resulting in several dependent variables. The maximum velocity attainable is a key value for understanding the range of values that must be processed by touch scrolling transfer functions, but we are also interested in various human and device characteristics that may help devices discriminate between user intentions for different scrolling velocities, including clutch time (or gesture frequency) and gesture length. The study factors, all within-subjects are as follows:

$$\textit{Speed} \in \{\textit{browsing, comfortably fast, maximally fast}\}$$
$$\textit{Device/Posture} \in \{\textit{iPhone-thumb, iPhone-finger, iPad}\}$$
$$\textit{Direction} \in \{\textit{up, down}\}$$

Device and *speed* were counter-balanced across subjects using a Latin Square; *direction* order was alternated.

Results and Analysis

Figure 4 uses box-whisker plots to show the cross-participant range, interquartile range, median, and mean values for three measures: maximum registered finger velocity (top), clutch time (middle), and gesture length (bottom).

Analysis of variance (with Greenhouse-Geisser sphericity corrections) of velocity shows significant main effects for *speed* ($F_{1.76,19.34} = 35.5, p < .001, \eta_p^2 = .76$), with means of 178, 399 and 560 mm/s at *browsing*, *comfortably fast* and *maximally fast*, respectively. The maximum velocity from any participant was 1378 mm/s with an upward gesture on the iPad. The main effect of *device/posture* was also significant ($F_{1.22,13.36} = 42.1, p < .001, \eta_p^2 = .80$), with thumb gestures on the iPhone slowest (mean 186 mm/s), then index-finger iPhone (390 mm/s) and iPad (560 mm/s). A significant *Speed×Device/Posture* interaction ($F_{2.60,28.57} = 18.5, p < .001, \eta_p^2 = .63$) is best attributed to the comparatively small difference between iPhone-thumb velocities across target speeds, as shown in Figure 4 (top). This interaction suggests that velocity may be a poor metric for discriminating between scrolling intentions—while it is likely to succeed for index-finger gestures, it could fail for one-handed thumb gestures. There was no effect of *direction* ($p = .05$), nor any other interaction.

Analysis of clutch time (Figure 4, middle) suggests that it is a fairly robust discriminator of the users' intention for different scrolling speeds. There was a significant effect of *speed* ($F_{1.30,14.25} = 16.4, p = .001, \eta_p^2 = .60$), with means of 536ms when *browsing*, 322 ms at *comfortably fast*, and 160 ms at *maximally fast*. *Direction* also showed a significant effect ($F_{1,11} = 5.6, p = .037, \eta_p^2 = .34$), with clutching after downward drags (mean 320 ms) slightly faster than upward ones (339 ms). No other main effects or interactions were significant. The results for clutch time suggest that users choose to increase the frequency of their gestures (reducing clutch time) when intending to scroll quickly.

Finally, analysis of gesture length showed that the device form-factor has a significant impact on gestures, with the larger size of the iPad encouraging much longer gestures (mean 92 mm) than the index finger or thumb on the iPhone (46 and 31 mm respectively)—$F_{1.39,15.31} = 75.12, p < .001, \eta_p^2 = .87$. There was no significant effect of *Speed* ($p = .35$), suggesting that gesture length is a poor metric for determining user intentions for different scroll velocities. No other effects were significant.

Observations of Subject Behaviour

Thumb gestures on the iPhone were expressed similarly, regardless of intended velocity, with interphalangeal articulation through contractor (down) or extensor muscles (up); explaining the relatively low variance in gesture length and velocity.

When using the index finger with the iPhone, several participants varied finger posture across velocity and direction. Approximately half of the participants used contractor-based interphalangeal articulation with the finger co-linear with the device to express downward gestures at *browsing* speed. Other participants used small amounts of finger contraction coupled with forearm movement to browse downward (with the finger at ~330° to device), and some used wrist ulna deviation with the finger substantially oriented across the device (~280°).

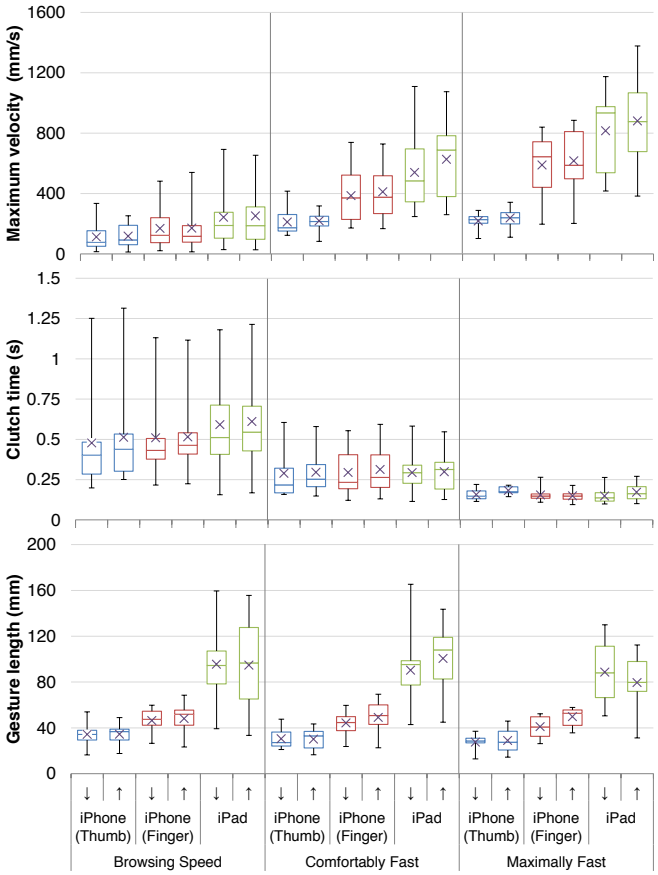


Figure 4. Box-whisker plots characterising the maximum velocity (top), clutch time (middle), and length (bottom) of gestures at three different target speeds with different device/posture combinations in up and down directions.

Upwards scrolling at browsing speed was less varied with most participants using radial deviation or lateral forearm movement with the finger oriented across the device. As the target speed increased to comfortably fast and maximally fast most participants switched to forearm-based movement with the finger pointing across the device. Broadly the same observations applied to the iPad, although few participants used interphalangeal articulation at browsing speed (using wrist or forearm movements throughout).

Discussion

These results provide insights into the parameters that people use to express their intentions for different scrolling velocities across different device form factors and postures. They also suggest the range of values that devices can expect users to express. We used the data from Study 1 to ensure that we probed a superset of human-expressible gestures in Study 2.

The primary finding is that clutch time is a reliable indicator of intended scrolling velocity, with users increasing gesture frequency to express a desire for faster scrolling. Gesture velocity also reliably differentiated between the intended scrolling velocities, but not when expressed with the thumb. Gesture length varied little between the intended velocities. One immediate implication of the clutch time finding is that transfer functions could allow accelerated scrolling by applying cumulative gain across repeated gestures.

For completeness we note some limitations in our method. First, as stated earlier, to focus on users' natural methods of gesture expression, we eliminated scrolling feedback. However, users will inevitably adapt their gestures to the output behaviour, and further work is needed to examine this relationship. Second, our method examined gesture data registered by the device, and quantization effects will introduce some error, particularly at high velocities. Regardless, transfer functions need to operate with registered data, so our results are indicative of what current devices can expect to receive. Third, physical properties of the device will influence gesture performance, including the surface material (e.g., glass vs. plastic), the existence of bezel edges or frames, and device size.

EXPOSING TOUCH SCROLLING TRANSFER FUNCTIONS

This section describes a variety of methods that can be used to examine touch-scrolling transfer functions, including the novel use of robotics to precisely emulate human movement.

Documentation and Source Code

Different vendors vary in their release of public information about scrolling. Google, for example, publishes the source code to their Android platform under an open source license, allowing open and complete inspection (however, device manufacturers can modify the open source code in a closed manner). Apple and Microsoft restrict public information to the API documentation about scrolling widgets (i.e., Apple's `UIScrollView` and Microsoft's `ScrollViewer` classes). This documentation describes properties that manipulate and can inspect limited aspects of the transfer function, but most details are absent.

Google Android

The following description highlights key elements of the transfer function in Android version 4.1 'Jelly Bean'.[1] Scrolling behaviour in widgets is built from several configurable components to perform input velocity tracking, gesture detection, and a deceleration function. To accommodate different device display densities, Android employs a measurement unit of 'density-independent pixels' (dp), where 1 dp is equivalent to one physical pixel on a 160 ppi (pixels per inch) screen. These values are scaled to the appropriate number of screen pixels for the display of a particular device.

The mechanics of these components is detailed in Study 2.

[1] `https://android.googlesource.com/platform/frameworks/base/+/jb-release/`

Simulators and Decompilation

When source code is unavailable, the vendor may provide device simulators or emulators that can be used to analyse behaviour. For example, Apple provides an iOS simulator that can be stimulated from other applications in the operating environment to simulate touch interaction. Furthermore, private APIs allow events recorded on the simulator to be replayed on an actual iOS device. Consequently, programmers can generate gestures on the simulator, edit the script describing the gestures (within certain constraints), and replay them on a device that is equipped to record the scrolling output resulting from the simulated gesture.

When a vendor does not provide tools supporting simulation or emulation, or if the reliability/fidelity of the tools is questionable, it is probable that the executable code can be disassembled and decompiled to reverse engineer a form of source code. For example, Microsoft ships the implementation of their scroll widgets in a virtual machine bytecode (CIL), which is amenable to disassembly.

Robotic Control

When simulators or emulators are unavailable, or if their fidelity is unknown, device behaviour is essentially a black-box, forcing analysts to guess how the device behaves based on unreliable observational methods. A major factor contributing to this problem is the lack of precision in human input control—the length, timing, speed, acceleration, direction, pressure, and other properties will vary substantially, regardless of attempts to control their consistency and accuracy.

To overcome this problem we used robotic control of touch contacts. SCARA robots (Figure 5) are particularly appropriate due to their high precision (±25 microns in three dimensions) and high maximum velocity (up to 11 m/s).

With programmable devices (such as phones and tablets) it is straightforward to write programs that log touch inputs and consequent scrolling output. However, when a device is non-programmable (e.g., an iPod Nano, which supports flick scrolling, but has no public methods for programming), it would be necessary to use high-speed video transcription to capture the scrolling output. It is important to log the contact data as well as the scrolling output in order to detect discrepancies between the intended robotic contact and that registered/processed by the device's touch sensor and firmware. For example, in this way we discovered discrepancies in the data describing a first contact after a pause of several seconds without contact (described earlier in Study 1).

STUDY 2: TRANSFER FUNCTION ANALYSIS AND RESULTS

We analysed the touch scrolling transfer functions supported by several contemporary devices using a variety of methods: Google Android functions using its source code (described earlier); Apple iPhone 4S and iPad 2 (both running iOS version 5.1.1) using the iOS Simulator on Mac OS X to emulate touch events that were subsequently recorded and replayed on actual

Figure 5. An an Epson G10-651S SCARA robot performing flicking gestures on an Apple iPad.

devices; and Apple iPhone 4S and iPad 2 using a SCARA robot. The iPhone and iPad were analysed using both robot and touch emulation methods to facilitate cross-validation of methods; any discrepancy in results would suggest errors or imprecision in either the robot method or in the touch emulation stemming from the iOS simulator.

Method

For robotic analysis we used an Epson G10-651S SCARA to manipulate a Wacom Bamboo Stylus (Figure 1). The Epson SCARA allows precise movements (±25 microns in three dimensions) at up to 8 m/s. Robot movement was programmatically controlled. Data describing contact information and resultant scrolling effect was logged on the device.

With robotic and emulated analysis (Apple devices), and with programmatic analysis (Android), we created a series of scripts to barrage the devices with batches of 16 repeated identical flick gestures. Each successive batch varied one input parameter at a time: *gesture length*, 10–160 mm; *gesture speed*, 5–200 cm/s; *clutch duration*, 100–1000 ms; and *direction*, up and down. We also examined other factors such as document length to rule out their influence. Multiple levels of each factor were initially investigated, and when the results indicated that threshold values were used to control transfer function behaviour, additional gesture-batch probes were created to successively hone in on the threshold values. After each gesture batch was complete, software logged any residual scrolling and then reset the scroll view for the next gesture batch.

Note that the set of parameters examined is not exhaustive, and other factors such as *acceleration*, *pressure*, and *contact area* may also influence behaviour. For expediency, we focused on the most likely parameters under default system settings.

Results

The Apple and Android behaviours maintain a 1:1 mapping between finger movement and output scrolling effect while the finger remains in contact with the display. Doing so is likely to maximise the user's perception of direct control of a physical surface while touching. Both systems also apply a 'slop' (Android) or 'hysteresis' (iOS) threshold that requires a minimum touch displacement before interpreting the input as a scrolling action (Android: 8 dp, iOS: 10 points[2]) to prevent accidental or noisy input channels. Once the finger leaves the surface, however, the transfer functions can substantially amplify the output scrolling effect (applying gain) to assist the user in traversing long distances. The conditions under which gain is applied, and the levels of gain, are described below.

Cross validation of the iOS transfer functions between the simulation and the robotic methods showed nearly identical results. The following subsections describe the primary parameters influencing the transfer functions (a complete set of data collected is also available[3]).

Flick Detection

Not every gesture is considered a 'flick' by the system; when the finger lifts from the screen, both systems examine recent touch information to classify the user's gesture as either a flick or a drag.

For Android, when the finger lifts off the display, up to the last 20 events over the last 100 ms are sampled and smoothed using the first order coefficient from a linear least squares solution to determine movement velocity. If this velocity exceeds a threshold of 50 dp, a 'fling' event is triggered. The velocity of the fling are used by a 'scroller' object to calculate the distance and duration (configurable with a coefficient of friction parameter).

On iOS, finger velocity is sampled between the movement events received. When the finger lifts from the touch surface, the velocity of two prior movement samples (v_t and v_{t-1}) are combined: $V_t = [(v_t + v_{t-1})/2] - [(v_t - v_{t-1})/4]$. If the resulting velocity is greater than 250 points per second, the gesture is classified as a 'flick'. A further transformation is then applied: $\frac{1}{4}V_t + \frac{3}{4}V_{t-1}$, which is used as the flick velocity for the gain and simulated friction calculations.

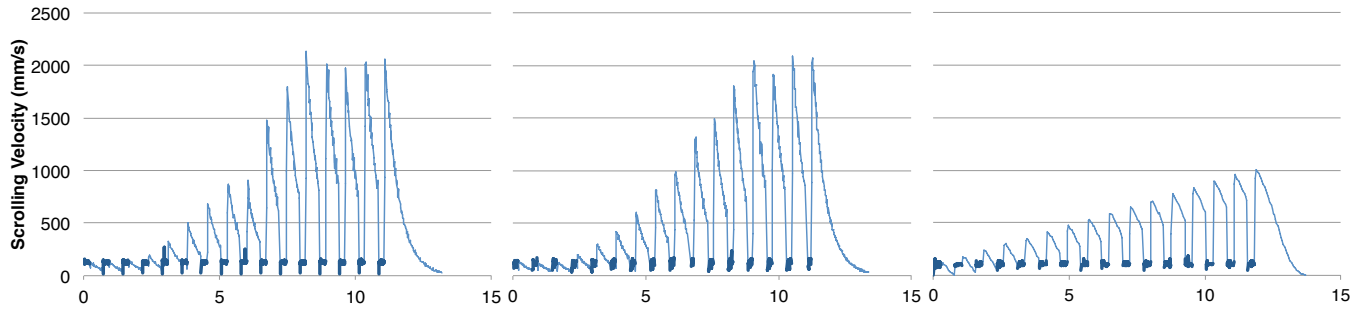
Cumulative Gain

Figure 6 shows the output scrolling velocity across time when a series of sixteen identical flick gestures (30 mm long at a velocity of 135 mm/s and with a clutching time of 500 ms) are issued on the touch surface—results are generated using iOS touch simulation (6a), iOS robotic control (6b), and Android simulation (6c). Thick lines in the figures show when contact is registered on the surface of the device. Note that the output velocity closely matches the velocity of the input control while surface contact is registered.

Both iOS and Android apply cumulative gain across gestures, but the manner in which they do so differs. With iOS, cumulative gain is applied as the finger lifts after the fourth repeated gesture, giving an instant acceleration to a higher velocity than that of the finger's movement. The scroll velocity is gradually

[2]The correspondence between points and physical pixels is determined by a device dependent 'scale factor' (either 1 or 2 on current non-'retina' and 'retina' devices, respectively).

[3]`http://cortex.p.gen.nz/reserach/scrolling`



(a) iOS Simulator. (b) iOS Robot. (c) Android Simulator.

Figure 6. Scrolling velocity across time (seconds) recorded following repeated gestures of 30 mm length, 135 mm/s velocity, and 500 ms clutch time using: (a) the iOS simulator, (b) iOS via. SCARA, and (c) Android code execution. Thick lines indicate touch contact.

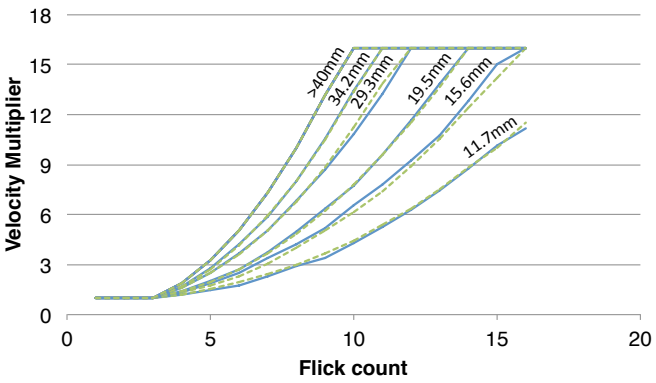


Figure 7. iOS (iPhone) velocity multiplier across successive flicks at different levels of gesture length and two levels of velocity (135 mm/s: dashed lines; and 300 mm/s: solid lines).

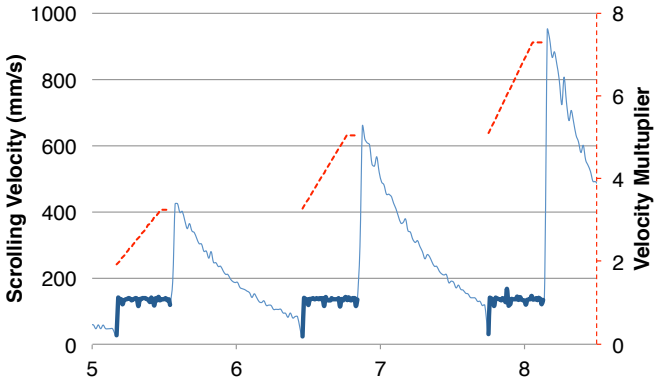


Figure 8. A detail of scrolling velocity from Figure 6a, shown with the value of the velocity multiplier (red, dashed line).

attenuated while contact is off the display, simulating friction. The iOS plots (Figures 6a and 6b) show that gain is capped; in this case after the 12th gesture.

When an Android Scroller's 'flywheel' function is enabled, any residual scroll velocity remaining during deceleration is added to the velocity of the next flick (Figure 6c). Unlike iOS, Android begins this amplification on the second flick and does not cap the maximum level of gain. Figure 6c shows scroll velocity gradually increases when receiving identical gestures to those used in Figures 6a and 6b. More precisely, Android approximates the actual residual scroll velocity based on the time spent since the previous flick gesture and the duration of contact for the current one, and adds it to the last flick velocity.

We found no evidence that the iOS transfer function attends directly to clutch duration, except for applying an ~900 ms timeout between gestures to be considered part of a gesture series. Android incorporates clutch time as a component of the deceleration effect.

Gesture Length and Direction

The cumulative gain applied across flicks on iOS is controlled by a multiplier on the flick velocity. Figure 7 shows how the value of this multiplier increases across successive flicks as the length and velocity of each flick increases. Only gesture length was found to influence the rate at which the multiplier increases, with longer gesture lengths (up to a maximum of ~40 mm) increasing the velocity the fastest. The multiplier was progressively raised until it reached a cap of 16×.

This behaviour can be seen in Figure 8, which shows detail of the scrolling velocity and the velocity multiplier across the fifth to seventh repeated gesture. Note that input and output velocity are identical while the finger is in contact, but the multiplier increases during this contact until release, when it is applied. More precisely, the value is incremented from the fourth contact onward by $\frac{1}{480}(k-1)$ for each point of finger movement (where k is the number flicks made in the series) until it reaches the cap. The cap is 1 before the fourth flick, 16 from the tenth onward, and $cap(k) = cap(k-1) + 0.45(k-1)$ for the intervening values.

Our analyses showed no difference in the application of gain across scroll direction, except that all methods reset cumulative gain across gestures when scroll direction changed.

Simulated Friction

Simulated friction effects slow the scroll velocity while the finger is out of contact with the surface. Figure 9 shows the

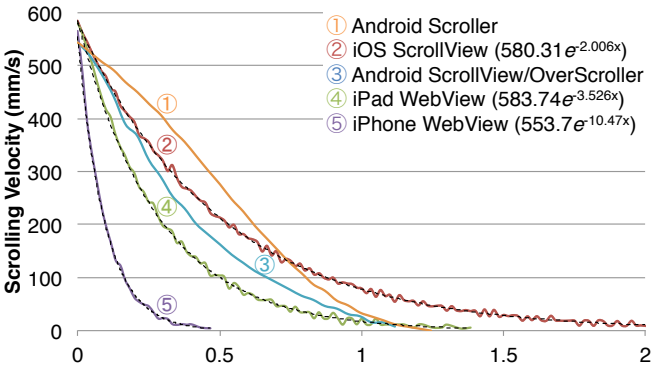


Figure 9. Friction/deceleration effects across time (seconds) following release on the iPad and iPhone with a `UIScrollView` (e.g., lists) and `UIWebView` (e.g., web browser), and the two scrollers provided on Android.

observed fall off curves with iOS, which vary with the type view displaying the content (`UIScrollView` for lists, and `UIWebView` for web content), and is configurable by developers (via. the `decelerationRate` property of `UIScrollView`). Exponential regression curves show accurate models of the observed behaviour, explaining more than 98% of the variance in all cases. `UIScrollView` effects are identical between the iPad and the iPhone; however, the `UIWebView` applies higher levels of friction on the iPhone.

Android's deceleration function depends on the 'scroller' used by the widget. Those that are provided by the system use either an `OverScroller` (scroll, list, and web view widgets) or a `Scroller` (gallery and text view widgets). Their behaviour can be customised, but the curves produced by their default parameters are shown in Figure 9. Regression curves did not produce accurate models of the observed behaviours; each class pre-computes a spline curve from piecewise functions.[4]

DISCUSSION

The results exposed the primary components of the iOS and Android touch scrolling transfer functions. We developed methods to analyse the transfer functions, conducted a human factors study to understand how users choose to control scroll velocity and to establish the range of values that should be used to probe the devices, and we deployed the methods to analyse the implementations of contemporary devices.

We revealed both substantial similarities and differences between iOS and Android. Similarities include: the maintenance of 1:1 correspondence between finger and scroll velocities while in contact; the application of cumulative gain across successive gestures; and progressive retardation of scroll velocity when the finger is out of contact. Differences include the iOS use of a capped scroll gain multiplier that is applied after the fourth successive flick gesture, while Android simply adds any residual scrolling velocity to the current gesture's velocity.

[4] `core/java/android/widget/OverScroller.java`, lines 606–637; `core/java/android/widget/Scroller.java`, lines 75–99.

Opportunities to Improve Scroll Transfer Functions

A primary reason for exposing these transfer functions is to facilitate their iterative refinement. Our results highlight three main opportunities for doing so, as follows.

In-Contact Gain. There are interesting opportunities in relaxing the 1:1 mapping of finger motion to scroll motion during contact. Figure 6 shows that the scroll velocity is continually slowed to that of the finger during contact. This is appropriate when scrolling slowly, because it helps maintain a sense of direct manipulation, but beyond some threshold it is unlikely that users will notice a breakdown of the mapping—allowing amplified scroll velocities throughout the drag and clutch operations. Implementing this relaxation would require careful design to ensure that a tap and hold gesture to stop scrolling was interpreted as such, at the user's expected scroll position.

Clutch-Time Dependent Gain. The human factors study suggested that users reduce clutch time (increasing gesture frequency) when they want to scroll faster, which could be explicitly incorporated as a parameter in gain functions.

Document-Length-Dependent Gain. Cockburn et al. [5] allows a 1:1 mapping at slow rates of input expression and a gain level that is determined by document length at high rates of input. This approach seems appealing for mobile devices that have small viewports and potentially very long views.

Facilitating Rigour in Touch Scrolling Studies

Another reason for exposing the transfer functions is to facilitate experimental rigour in touch scrolling studies. The methods described in this paper allow researchers to establish the functions used in any system used for baseline comparison. We recommend that future researchers examining touch scrolling systems report at least the following components of their experimental transfer functions: (1) the mapping used while the finger is contact with the display; (2) the function used to apply gain; (3) the deceleration function; and (4) the device resolution, and how physical touch movement is mapped to display pixels.

We are also releasing the data tables collected in our study, revealing the range of touch input we tested and the complete system response, and the source code to several tools used in gathering this data. These tools and data are available at: `http://cortex.p.gen.nz/reserach/scrolling`.

Limitations of Robotic Testing

SCARA robots provide a method of controlling and testing touch sensitive devices that do not allow direct programmatic control. While they allow extremely accurate control over the timing and location of discrete contacts, our experience suggests their velocity control is less stable. Our model allowed two modes of control, 'Continuous Path' (CP) and 'Point-to-Point' (PTP). In PTP mode the arm stops at each specified point in (x, y, z) space, while in CP it interpolates between points and attempts to maintain a constant velocity. Although it allows extremely rapid acceleration (up to 5 m/s^2, rocking

its 2 tonne concrete mounting block), the resultant velocities logged on the device were not as constant as desired (see Figure 6b). Despite this limitation, robotic analysis revealed similar results to those generated with the simulator. It is worth noting that for closed devices, such as the Apple Magic Trackpad, robot control may be essential to analyse behaviour.

CONCLUSIONS

Touch scrolling is a fundamental component of mobile interaction, yet there has been little public research into how current methods achieve their behaviour. This is a problem for researchers, as they lack information on which to base iterative improvements, and experimental comparisons against contemporary systems may be confounded by failing to attend to a key parameter embedded in closed, proprietary systems. We have presented methods that can be used to reveal the parameters employed by touch scrolling transfer functions, and used the results of a human factors study to inform an examination of Apple iOS and Google Android touch scrolling transfer functions. We found interesting properties in both the human interaction features that change with a user's scrolling intentions, and the properties of touch input that are attended to by the system. These methods and results, together with the data tables and software accompanying the paper, provide a firmer foundation for research seeking to improve touch scrolling transfer functions and validate their effectiveness.

ACKNOWLEDGEMENTS

This work was supported by a Royal Society of New Zealand Marsden Grant 10-UOC-020.

REFERENCES

1. Aliakseyeu, D., Irani, P., Lucero, A., and Subramanian, S. Multi-flick: an evaluation of flick-based scrolling techniques for pen interfaces. In *Proc. CHI '08*, ACM (2008), 1689–1698.
2. Ausbeck, Jr., P. J., US. Patent No. 7446754. U.S. Patent and Trademark Office (Washington, DC, 2008).
3. Baglioni, M., Malacria, S., Lecolinet, E., and Guiard, Y. Flick-and-brake: finger control over inertial/sustained scroll motion. In *Proc. CHI EA '11*, ACM (2011), 2281–2286.
4. Casiez, G., and Roussel, N. No more bricolage! Methods and tools to characterize, replicate and compare pointing transfer functions. In *Proc. UIST '11*, ACM (2011), 603–614.
5. Cockburn, A., Quinn, P., Gutwin, C., and Fitchett, S. Improving scrolling devices with document length dependent gain. In *Proc. CHI '12*, ACM (2012), 267–276.
6. Harrison, C., Schwarz, J., and Hudson, S. E. Tapsense: enhancing finger interaction on touch surfaces. In *Proc. UIST '11*, ACM (2011), 627–636.
7. Hinckley, K., Cutrell, E., Bathiche, S., and Muss, T. Quantitative analysis of scrolling techniques. In *Proc. CHI '02*, ACM (2002), 65–72.
8. Hinckley, K., and Wigdor, D. Input technologies and techniques. In *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications*, J. A. Jacko, Ed., 3rd ed. CRC Press, 2012, ch. 9.
9. Hinckley, K. P., Bathiche, S. N., Cauthorn, J. H., and Sinclair, M. J., U.S. Patent No. 6690365. U.S. Patent and Trademark Office (Washington, DC, 2004).
10. Hinckley, K. P., and Cutrell, E. B., U.S. Patent No. 7173637. U.S. Patent and Trademark Office (Washington, DC, 2007).
11. Johnson, J. A. A comparison of user interfaces for panning on a touch-controlled display. In *Proc. CHI '95*, ACM Press/Addison-Wesley (1995), 218–225.
12. Kwon, S., Choi, E., and Chung, M. K. Effect of control-to-display gain and movement direction of information spaces on the usability of navigation on small touch-screen interfaces using tap-n-drag. *Int. J. Ind. Ergonomics 41*, 3 (May 2011), 322–330.
13. Lee, J., Lee, D., and Chung, M. K. Evaluation of mapping functions for one-handed flick operations on a mobile device. In *Proc. MobileHCI '11*, ACM (2011), 123–131.
14. Malacria, S., Lecolinet, E., and Guiard, Y. Clutch-free panning and integrated pan-zoom control on touch-sensitive surfaces: the cyclostar approach. In *Proc. CHI '10*, ACM (2010), 2615–2624.
15. Ording, B., Forstall, S., Christie, G., Lemay, S. O., and Chaudhri, I., U.S. Patent No. 7786975. U.S. Patent and Trademark Office (Washington, DC, 2010).
16. Quinn, P., and Cockburn, A. Zoofing! Faster list selections with pressure-zoom-flick-scrolling. In *Proc. OZCHI '09*, ACM (2009), 185–192.
17. Quinn, P., Cockburn, A., Casiez, G., Roussel, N., and Gutwin, C. Exposing and understanding scrolling transfer functions In *Proc. UIST '12*, ACM (2012), 341–350.
18. Smith, G. M., and schraefel, m. c. The radial scroll tool: scrolling support for stylus- or touch-based document navigation. In *Proc. UIST '04*, ACM (2004), 53–56.
19. Tu, H., Wang, F., Tian, F., and Ren, X. A comparison of flick and ring document scrolling in touch-based mobile phones. In *Proc. APCHI '12*, ACM (2012), 29–34.
20. Zhai, S., and Smith, B. A. Multistream input: an experimental study of document scrolling methods. *IBM Systems Journal 38*, 4 (1999), 642–651.
21. Zimmerman, J., and Martino, J. A., U.S. Patent No. 6690387. U.S. Patent and Trademark Office (Washington, DC, 2004).