Hammer Time! A Low-Cost, High Precision, High Accuracy Tool to Measure the Latency of Touchscreen Devices

Jonathan Deber[1,3] Bruno Araujo[3,1] Ricardo Jota[1,3] Clifton Forlines[1,3]
Darren Leigh[2] Steven Sanders[2] Daniel Wigdor[3]

[1] Tactual Labs
Toronto, ON, Canada
{first.last}@tactuallabs.com

[2] Tactual Labs
New York, NY, USA
{first.last}@tactuallabs.com

[3] University of Toronto
Toronto, ON, Canada
{brar, daniel}@dgp.toronto.edu

ABSTRACT

We report on the Latency Hammer, a low-cost yet high-accuracy and high-precision automated tool that measures the interface latency of touchscreen devices. The Hammer directly measures latency by triggering a capacitive touch event on a device using an electrically actuated touch simulator, and a photo sensor to monitor the screen for a visual response. This allows us to measure the full end-to-end latency of a touchscreen system exactly as it would be experienced by a user. The Hammer does not require human interaction to perform a measurement, enabling the acquisition of large datasets. We present the operating principles of the Hammer, and discuss its design and construction; full design documents are available online. We also present a series of tools and equipment that were built to assess and validate the performance of the Hammer, and demonstrate that it provides reliable latency measurements.

Author Keywords

Input latency; multi-touch input; latency measurement; latency benchmark

INTRODUCTION

I often say that when you can measure what you are speaking about and express it in numbers you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind…

− William Thomson, 1st Baron Kelvin [23]

The impact of a computing system's interface latency, the time between a user's action and the system's response, is a central question in the development of interactive computing systems and has been studied since at least the 1960s [2, 8, 9, 13, 15]. Some degree of latency is an inherent part of any system, since a variety of tasks must be performed in order to process a user's action: the state of the input device(s) must be sampled, computations performed, graphics generated, and displays updated. Ideally, the latency would be kept below the threshold that can be detected by the human visual system, which would render the interface indistinguishable from a truly latency-free system. Existing commercial touchscreen devices have latencies that range between 50 and 200 ms [15], and numerous researchers have demonstrated that this latency is quite noticeable [2, 7, 8, 10, 15]; indeed, some research has suggested that humans can perceive touchscreen latency as low as 2 ms [15]. There are also performance improvements for dragging tasks on touchscreens if latency is reduced below 25 ms [8].

Clearly, latency reduction is a desirable goal and is the focus of much work in both academic and industrial settings. A critical part of any effort to reduce latency is the ability to measure it, since without measurement we cannot assess the results of our efforts [23].

While many systems exist to measure latency [4, 6, 15, 21], they tend to suffer from one or more of the following issues: they approximate the latency measure or are imprecise; they are cumbersome and require expensive hardware (e.g., high-speed cameras, robotic arms); or they require human interaction making the measurements subject to human imprecision. While these limitations may have been acceptable for measuring latencies on the order of 100 ms, more recent work has demonstrated technologies which are capable of reducing latency far below these values [11, 15]. Thus, we argue that there is the need for inexpensive test equipment that can directly and empirically measure the interface latency of a touchscreen device without those issues. We set out to build such a system, which we call the Latency Hammer. Our prototype is shown in Figure 1. The design is open source; full design documents are available at http://www.tactuallabs.com/latencyhammer/.

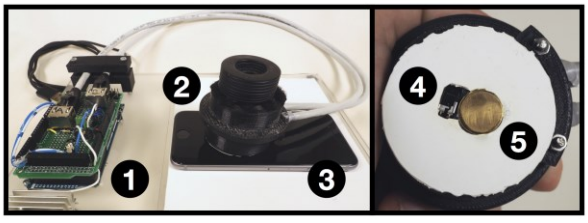

Figure 1. The Hammer prototype is composed of a microcontroller (1) and a measurement head (2) that rests on the device under test (3). The measurement head contains a photodiode (4) to measure display brightness and a brass contact (5) to trigger touch events.

CHI'16, May 07 – 12, 2016, San Jose, CA, USA

ACM 978-1-4503-3362-7/16/05… $15.00
DOI: http://dx.doi.org/10.1145/2858036.2858394

The Hammer is designed to meet the following goals:

Precision & Accuracy: Given the lower bounds of human latency perception, a measuring tool should be accurate and repeatable to within 1 ms.

Range: Some commercially available systems have latencies well over 100 ms, while some research-based systems have latencies of less than 1 ms. A measuring tool should be able to report a wide range of latencies.

Cost: The required equipment and infrastructure should be inexpensive, ideally costing less than $100 USD.

Flexibility: The tool should be able to measure any capacitive touchscreen device, regardless of size or platform.

Invasiveness: The tool should be able to measure devices whether or not the tester has the ability to change or instrument the operating system, or even the ability to install application software on the device.

Automatic Operation: The tool should be capable of taking measurements without any human involvement after the initial setup and should be able to take a large number of unattended measurements within a short timespan in order to generate statistically significant results.

In this paper we report on the design and verification of the Latency Hammer, and how it meets our goals. We first report on previous work that has looked at understanding and measuring latency. We then introduce the theory of operation that is the basis for our current prototype. This is followed by a detailed discussion of the implementation, as well as the multiple validation procedures we undertook to verify that the Hammer was accurate down to the millisecond, including the design and construction of testing equipment for the Hammer itself. Finally we report on the usage of the Hammer in real-world scenarios and the complexities involved with measuring latency in commercial devices.

RELATED WORK

We focus our review of related work on two areas. First, we examine studies of human perception of and performance under latency on direct-touch systems, which demonstrate the value of reducing latency. Second, we examine past efforts to measure end-to-end latency.

Perception of Latency for Touch Input

Latency for interactive systems has recently been the subject of significant research [2, 7, 8, 15]. The overall findings indicate that user interaction is affected by latency at levels well under the ~50 ms observed in today's fastest devices [15]. Previous work has shown that it is harder for a user to detect the latency of a system's response to indirect input than the latency in response to direct-touch input [7]. This is not unexpected, since, as Deber et al. observe [7], the user's finger acts as a visual referent for when and where the system's output should appear with zero latency, offering a clear basis for comparison. Our work focus on direct-touch.

Kaaresoja et al. studied the visual perception of a physical button activation [9]. They found that the lower threshold of perception when pressing a physical button is 85 ms, and that latencies above 100 ms significantly affect the perceived quality of the interaction.

Recent advances in ultra-low latency direct-touch devices—with latencies of less than 1 ms—have enabled a series of studies that have explored the limits of latency perception [7, 8, 15]. Ng et al. studied the lower limit of perception for touchscreen dragging and found that users could distinguish as little as 6 ms of latency when comparing against a 1 ms referent [15]. Jota et al. studied the perception of tapping actions on touchscreens, complementing Ng et al.'s previous contribution [8]. They found that users are unable to discern latencies below 24 ms. Deber et al. studied direct and indirect input, and showed that small improvements in latency from a starting baseline are still perceived by users across most baseline latencies [7]. Latency reductions as low as 8.3 ms were noticeable when dragging on a touchscreen.

Latency perception has also been studied for direct input with a stylus [2, 13]. Ng et al. report that 2 ms is discernible for dragging tasks and that 6 ms is noticeable for scribbling tasks [13]. In a follow up study, Annett et al. introduced the complex tasks of writing and drawing [2]. They found a higher perceivable value of 50 ms, but argued that the complexity of the task affects perception.

These previous works strongly suggest that latency of systems should be lower than the current commercial values of 50–200 ms, driving our design requirement that the Hammer be capable of measuring latencies as low as 1 ms.

Measuring Latency and Triggering Touch

Other researchers have developed touch simulators for capacitive screens, although they have not been used to measure latency. For example, Yu et al.'s TUIC-*f* tangible tags had capacitive contacts that could be switched on and off; individual tags were identified by switching them at distinct frequencies [25].

Many early approaches to measuring interface latency used external cameras to calculate the time difference between an input and the corresponding visual response. Steed [21] mounted a tracker on a pendulum and used this as an input device that followed a known sine curve generated by the pendulum's motion. Using an external camera he captured the pendulum's movement and corresponding device output, converted each movement to sine curves, and compared the phase difference to calculate the latency. However, this approach cannot be directly applied to direct-touch devices.

Casiez et al. demonstrated how a standard optical mouse could be used to measure latency in a mouse-based UI [5]. The mouse was placed on top of the screen, which displayed a moving pattern. This motion fooled the mouse's optical sensor into thinking that the mouse had physically moved, and allowed a latency calculation based on the interval between the change in the pattern and the cursor movement. While effective for mouse-based systems, this technique is not suitable for a touchscreen-based device.

Other approaches have measured physical distances during dragging operations. Ng et al. [15] used a high-speed camera to capture finger motion when dragging. The drag was performed at a constant speed on a known path and used a physical ruler to constrain the movement and aid the distance calculations. The physical distance between the finger and

the response was used to calculate the latency. Some commercial systems use extremely precise industrial robots coupled with high-end high-speed cameras [18], but these systems are prohibitively expensive for most users.

Bérard and Blanch proposed two methods of measuring latency [4]. The first, a high precision method, uses a camera to track a finger's position with an added marker, and carefully synchronizes those images with the trajectory of the finger as recorded by the touchscreen. This approach is able to calculate the system's latency with a precision of 2 ms. The second approach is less precise but does not require external hardware. An on-screen target is moved in a circular path and the user is asked to follow it with their finger. Since the finger position should correspond to the currently displayed position of the target, the distance between the target's expected and actual on-screen positions can be used to calculate the latency. While useful, both of these techniques require intense human interaction, with the second method being particularly prone to operator errors, making it only precise to 4 ms.

Cattan and Bérard [6] built on Bérard and Blanch's second method by using latency prediction to simplify the task of aligning the finger with the target. A predictive algorithm is fine tuned to the system's visual response in order to reduce the apparent latency of the response to a finger. Once the results are satisfactory, the system latency can be obtained by looking at how the predictive algorithm was parameterized.

Liang et al. [12] demonstrated a latency estimator for a head mounted display based on a camera recording the system's graphical feedback. The input device (a Polhemus Isotrak) is attached to a pendulum with a known motion, and latency is computed from the estimated spatial gap and pendulum speed. Others have demonstrated equivalent solutions using other motions. Ware and Balakrishnan [24] adopted a similar approach but used a stepper motor to generate linear motion instead of a pendulum. Swindell et al. [22] utilized a turntable to move the input device in a circular motion. Pavlovych et al. [18] were able to measure latency with an unconstrained trajectory. They use a camera to record a mouse movement and the corresponding cursor movement, and compare the two video feeds to obtain the time between a mouse reaching a target and the corresponding cursor reaching the same location. While reporting results, the authors indicated that the measurements are affected by the inaccuracy of the apparatus (e.g., video-based motion tracking).

While these methods provide latency measures, they require human motion for each measurement and can thus introduce a source of error. We believe that tools that enable fast and efficient unattended batch collection of latency measurements are crucial to further our attempts to reduce latency.

THEORY OF OPERATION

The goal of the Latency Hammer is to provide a reliable apparatus to accurately measure the end-to-end interface latency associated with tapping on a touchscreen. This means measuring the time from touch activation to the display of the corresponding visual response. This interval includes both software and hardware components: the capacitive sensor detects the touch; the touch information is captured by the hardware and processed by the OS; a message is passed to the active application; the application updates its internal state and draws new information to a display buffer; the display buffer is passed to the operating system; the operating system asks the display to render the display buffer; the display updates its pixels, and is able to finally show the user the feedback for the touch activation. Within this flow, each step contributes to the final end-to-end latency.

However, latency is not constant, and some variability is to be expected. While the input sensor and the display are scanned and updated cyclically (typically between 60 and 120 Hz, with constant data transfer costs), the touch sensing is not always synchronized with the finger landing on the surface. In addition, the system load is not always consistent, and the display is not always ready to render when a buffer swap is requested. Moreover, latency is strongly connected to the display refresh rate. On a 60 Hz display, new frames are drawn every 16.7 ms. Minor improvements in latency may be absorbed by the refresh rate, and large increases in latency may suddenly occur if a small additional delay forces the graphical update to wait for the next display frame.

Although it is tempting to attempt latency measurements based on timestamp information internal to a device, both software and hardware factors affecting performance mean that the most accurate measurement of latency can only be taken by 1) optically observing the screen's response to a physical input, and 2) running a series of trials to cope with measurement variations, as described above. We therefore focused our efforts on developing a device that could measure end-to-end latency—from activation to observable visual response—and that could automate multiple measurement cycles, reducing human intervention to its bare minimum (i.e., initial setup for the measurement session).

The underlying concept of the Latency Hammer is straightforward: the Hammer is placed on top of a touchscreen device running an application that responds to a touch event by causing a significant change in brightness. Prior to each measurement session, the Hammer measures the light levels associated with the bright "on" and dark "off" states of the screen and calibrates brightness thresholds for each state. A host computer controls the Hammer, and an operator can initiate multiple runs of unattended measurements. The latency measurements are automatically clustered and analyzed; the results are provided to the operator in both graphical and textual formats.

When a measurement cycle is initiated, the Hammer records a timestamp and then uses a touch simulator to trigger the capacitive screen and generate a touch event. This is entirely automated and does not involve a human, although from the touchscreen's point of view the simulated touch is indistinguishable from a human tapping the screen. After initiating the touch, the Hammer uses a photo sensor to watch for the change in screen brightness caused by the response to the touch. Once a change is observed, the Hammer records a second timestamp. The elapsed time between the timestamps constitutes a single latency measurement. The automated nature of the Hammer means that hundreds of measurements can easily be collected when evaluating a device.

To maximize changes in brightness, ideally we make use of one of our custom Hammer applications; these have a black screen and then draw a white region in response to a touch event. However, it is possible to use any change in UI brightness (e.g., a button activation highlight) at the cost of some sensitivity. This allows us to use the Hammer to measure the latency of commercial applications and on devices that do not permit third-party code.

The version of the Latency Hammer described in this paper has no moving parts in the touch simulator. Although this presented some design challenges, detailed in the following section, solving these issues allowed us to reproducibly and precisely trigger touch events without human intervention. Removing the human element from the measurement process enabled us to automate it; we believe that this is a significant contribution of the Hammer prototype.

Figure 2 presents a diagram of the main components of the Hammer. The measurement head contains the touch actuator (a brass contact approximately the size of a human fingertip) and brightness sensor (a photodiode). When positioned on top of a capacitive touchscreen, the brass contact is designed so that it will not trigger a touch by itself. However, if it is electrically connected to an electron sink that provides additional mass and surface area, it will trigger a touch event. An electronically actuated normally open (NO) switch controls this connection. When the switch is closed, the brass contact is connected to the electron sink, disturbing the capacitive levels on the touchscreen and triggering a touch.

HARDWARE IMPLEMENTATION

Implementing the Hammer requires us to provide solutions for every step defined in the previous section: generating a touch event, capturing feedback from the display, and accurately calculating the time between the two events. Moreover, there are some practical considerations (e.g., securing the Hammer to the device) that influence the design.

In this section, we describe the implementation of our prototype. We begin with a discussion of the measurement head (responsible for generating touches and sensing the screen's response) and close with a discussion of the main board, which houses the microcontroller. Detailed construction details are provided online.

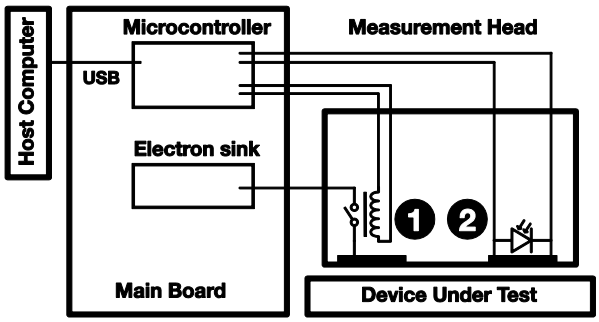

Figure 2. Block diagram of the Latency Hammer. The main board triggers a touch by closing the switch (1) to generate a capacitive disturbance on the screen. The visual response is monitored by the photodiode (2) connected to the microcontroller. The computed latency is then output to the host computer.

Measurement Head

The measurement head is a custom 3D printed roughly cylindrical enclosure that includes the circuitry to generate the simulated touch and deliver it to the screen through a brass contact, as well as a photodiode to sense feedback (see Figure 1, right). The brass contact and photodiode protrude through a hole in the bottom of the head, and rest on the screen during testing. The remainder of the bottom surface is covered in a layer of EPDM rubber to allow the head to grip the screen. A small circuit board is mounted inside the head and contains the switch (and supporting circuitry) that generates the touch; this switch is discussed in detail in the following section. The top of the head is covered in Velcro hook-and-loop fastener which enables repositionable weights that help the measurement head sit flat on the screen. We now detail the two main functions of the measurement head: generating touch and sensing feedback.

Generating Touch

One of the goals of the Hammer was to reduce the possibility of human-induced errors during the measurement. To effectively remove the human from the measurement loop the Hammer must create a simulated touch that is indistinguishable from an actual human touch. In order to do this it generates a change in the screen capacitance that the sensor identifies as a touch, and it is able to programmatically trigger and reverse this change.

Generating changes in capacitance: To generate these changes in capacitance, we use a brass contact connected to an electron sink. Brass was selected because it is highly conductive, corrosion resistant, and relatively soft (as metals go) reducing the likelihood of screen damage due to scratches; it is also relatively lightweight which reduces the likelihood of screen damage due to impacts. The contact is 12 mm in diameter (similar to a fingertip) and 2 mm tall. The design parameters for the contact are quite flexible but care must be taken to minimize the surface area so the contact does not trigger a touch event on its own. The inside of the brass contact (i.e., the side that is facing into the measurement head enclosure) was coated in liquid rubber insulation.

The brass contact is connected to an electron sink in order to dissipate additional charge. It is actually quite easy to dissipate sufficient charge, and in most cases the greater challenge is preventing an unintentional dissipation. As an example, connecting the brass contact to one end of a 30 cm 22 AWG wire with alligator clips (leaving the other end of the wire disconnected) was enough to induce a touch event. The current prototype repurposes a small bare aluminum heat sink (Ohmite R2V-CT6-38E [16]), since the characteristics of a heat sink (conductive material arranged to maximize its surface area) overlap completely with the design requirements of an electron sink. By connecting the contact to the electron sink, we could reliably trigger a touch. We next needed a way to programmatically enable and disable that connection.

Triggering touch programmatically: The choice of the electrically actuated switch is a critical part of the Hammer's design because a very small change in capacitance (on the order of 1 pF) is needed to register a touch event on a capacitance sensor and most switches leak too much current when open. This leakage would result in a touch event as soon as the Hammer came in contact with the screen, regardless of the switch position. During the development process numerous alternatives were examined and tested, including solid-state analog switches, tri-state devices, transistors, solid-state relays, and mechanical relays. We saw the most consistent performance with an Omron G6J-2P-Y, which is a highly insulated mechanical relay [16]. This relay, along with careful hardware design, was able to completely eliminate erroneous touch events.

Despite its advantages, a reliance on a mechanical switch introduced new problems. Switching a mechanical relay involves energizing a coil to magnetically move a metal contact, and a movement in the macroscopic world takes far longer—on the order of several ms—than the sub-atomic equivalent in a solid-state switch. While we could safely ignore a few ns of solid-state switching time when taking ms-scale latency measurements, we cannot ignore a ms-scale switching time. Fortunately, there was a straightforward way to measure the switching time in real-time during each latency measurement: we used a double pole relay with two switches controlled by a single coil. One pole was used to switch the touch simulator, and the other pole was fed into an input pin on our microcontroller. A change on that pin indicated the relay had closed, allowing us to subtract the relay's switching time from the latency measurement. A related concern is the bounce associated with the closure of a mechanical switch. Physical contacts can vibrate, causing a short period of rapid oscillations when a switch closes. We measured the bounce with an oscilloscope, and over a dataset of 50 relay closings found an average bounce time of 0.118 ms (SD=0.003), which could safely be ignored.

Finally, when designing the circuitry to trigger touch, special attention was paid in order to minimize current leakage and stray capacitance, since they could trigger unintended touches. Because the relay is acting as a switch between the brass contact (which is always resting on the screen) and the electron sink, one half of the switch will always be connected to the brass contact, regardless of the switch position. In order to minimize the length of the connection, and therefore reduce leakage, the relay must be placed as close as possible to the brass contact in the measurement head, which necessitates a small circuit board inside the head.

Sensing Feedback

The Hammer must be able to identify feedback generated by the simulated touch in order to compute the device's latency. Any screen response can be identified as a change in brightness, ideally as a transition from a black screen to a white one. We use a Vishay BPW46 PIN photodiode [20] to identify changes in brightness. This photodiode was selected because it has a fast and consistent response time, a flat-sided package that can sit flush against the screen, and good visible light sensitivity.

The photodiode is mounted in the measurement head adjacent to the brass contact, which allows it to observe a response in close proximity to the touch event, and is connected to an ADC (analog to digital converter) in the microcontroller via a shielded cable to the main board.

Because we run a raw analog signal over the cable, it is somewhat sensitive to noise. In the current design, we do not have any active circuitry adjacent to the photodiode in the measurement head, since the size of circuit that could be accommodated was limited. Placing a high-speed ADC in the measurement head would allow us to transmit a digital signal from the measurement head to the main board, and that signal would be more resilient to noise. However, the current approach is quite workable, and we have been able to mitigate most noise issues using both hardware (e.g., careful shielding) and software (e.g., detecting anomalous photodiode readings and rejecting any compromised trials).

Main Board

With the brass contact and photodiode in place, we now describe the core of the Hammer. The main board contains the microcontroller and supporting circuitry; it is responsible for triggering the switch to generate a touch, and then sensing the photodiode for changes in brightness. Once this is detected, the Hammer can calculate the time difference and output a latency measurement.

The brain of the Hammer is an Arduino Due microcontroller board [3], which is based on an Atmel SAM3X8E ARM Cortex-M3 CPU running at 84 MHz. In addition to the digital I/O pins, we make use of one of the built-in 12-bit ADCs to read the photodiode light levels. The customized supporting circuitry for the Hammer is built on a protoshield, which provides 30 cm^2 of perfboard in an Arduino-compatible form factor. The measurement head is connected to the main board via a pair of shielded Category 6 STP cables (i.e., Ethernet cable and connectors). The cables do not carry Ethernet signals; they are simply readily available shielded multi-conductor cables with built-in connectors. Once the Hammer has a measurement, it is sent to the host computer, which aggregates and logs all the measurements. The software for the microcontroller, the host computer, and device-specific helper applications are described in the next section.

HAMMER SOFTWARE

There are several software components that are required to perform a latency measurement. The main responsibility of the software is to coordinate the events defined in the previous sections and to store that information in the host computer for later analysis. Software is divided in two main components: the firmware, and the host computer software. There is also software that runs on the device under test. We provide "white touch" applications for several platforms that draw a black screen and then respond to a touch event by drawing a white region, but any application which causes a brightness change at the point of a touch will suffice.

Firmware

The firmware is responsible for the low-level actions involved in running a latency measurement and for transmitting the results back to the host computer. It can also run a series of precisely spaced measurements, which is used by the host software to create extended latency datasets; the data flow is described in the following section. Finally, the firmware is also responsible for maintaining a calibration of brightness levels of the "on" and "off" touch states. Written in the Arduino variant of C, the firmware interacts with the host computer via a USB-based serial console. The protocol is human-readable to facilitate easy debugging operations.

Host Computer Software

The host software is the front end of the Hammer. It is an interactive command line application written in Python that sends commands to the firmware, receives and analyses the results, and outputs data files and charts to the user.

Dataflow

A sample dataflow is as follows: The user places the Hammer on the screen of the device under test, and indicates the desired number of series to run, each of which consists of 10 measurements. Extended testing could use 100 series (for a total of 1000 measurements), but a small number of series (e.g., 5) will generally yield a good overview of the device's latency. The host software instructs the firmware to calibrate the "on" and "off" light levels and then begins running the first series of tests. Between each series, a randomized delay between 0–16.7 ms is communicated to the Hammer in order to distribute the measurements across the screen's refresh cycle. When all series are complete, the user is alerted, and results are shown on-screen. During the capture of latency measurements, user interaction is only required if the Hammer has detected a malfunction (e.g., a human is required to push a power button to wake the device).

Once all of the data is collected, the host software runs a k-means clustering algorithm across all of the measurements from all of the series. This is done because we expect groupings based on the display refresh cycle. The output data typically cluster around multiples of the display refresh time. The raw and clustered data is then logged, and a visual representation of the observed latencies is displayed. A sample chart is shown in Figure 3. Raw data is also made available to the user for logging or further analysis.

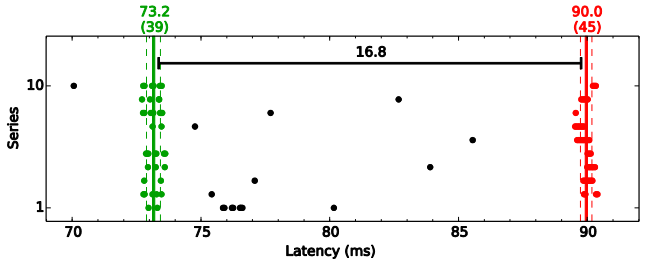

Figure 3. Graphical summary generated by the Hammer showing 100 measurements of a Nexus 9. The dataset contains 10 series, each of which contains 10 measurements. Each row corresponds to one series. Measurements cluster into two groups, with means spaced one display frame (16.8 ms) apart, illustrating the variability of the device's latency.

VALIDATION

The development of any type of test equipment requires a careful and thorough validation process to ensure that the equipment is, in fact, measuring reality. In many cases it is possible to verify new test equipment against an existing gold standard or test regime, providing a chain of metrological traceability. However, there is no standard or regime for latency measurement devices. We therefore contribute to the community by developing a series of tests and test equipment to provide confidence that the Hammer is correctly measuring latency. We hope to contribute towards a gold standard with our prototype and with our set of tests geared towards validating future measuring devices.

Our first test measures relative latency. We intentionally add latency to a device, and check that the Hammer is able to detect it. The second test measures the overhead involved in a Hammer measurement by creating and measuring a zero-latency device. The third test seeks to validate the absolute accuracy and repeatability by creating a simple oscillator-based latency generator circuit and simultaneously measuring it with the Hammer and with an oscilloscope. The fourth and final test cross-validates the Hammer by building a camera-based apparatus and simultaneously measuring a real device with both the camera and the Hammer.

Software Delay Test

We created two variants of our "white touch" applications (running on iOS and Android) that added an arbitrary delay prior to drawing the white region in response to a touch. Each application added this delay with a different programmatic mechanism. As our first measurement of the Hammer's accuracy, we attempted to measure that added latency. This allowed us to verify the relative measurement performance of the Hammer (e.g., configurations A and B are 50 ms apart) independent of the Hammer's absolute measurement performance (e.g., configuration A has 100 ms of latency).

The first application introduces milliseconds of latency (using POSIX `usleep`). We then ran several series of latency measurements using different added latencies using an iPad mini running iOS 7.1.2. The second application delays the response by a certain number of display frames, rather than by a certain number of ms. This allowed us to take into account the device's limitation in rendering to the display and measure latency in display frames. To add additional robustness to our verification, we opted to use Android for this test, and used Android's Choreographer to schedule the screen update in the future. Tests were performed on a Nexus 9 running Android 5.1.1. Table 1 shows the results for both iPad and Nexus 9 tests.

Overhead Test

The Hammer is only useful if it can measure absolute levels of latency. A key issue in calibrating these measurements is to understand the overhead of the measurement process. Several steps in the measurement process have small but finite durations (e.g., the time required for the ADC to read

the photodiode's signal), and we must ensure that these steps do not introduce an uncorrected delay that would skew the latency measurements. To measure this overhead, we created a device configuration that mimicked a zero-latency device by simply displaying a white screen. This meant that the Hammer would detect a visual touch response as soon as it starts checking for one, and any reported non-zero latency would therefore entirely consist of measurement overhead.

After 20 sample measurements, we found that the overhead was a consistent 0.006 ms (SD = 0 ms), which is several orders of magnitude smaller than the quantities that we are measuring, confirming that the Hammer overhead can be safely ignored in our latency calculations.

Oscillator Test

Having established the non-effect of overhead, we wished to measure the accuracy of absolute latency testing. Because commercial devices exhibit significant variability in their latencies between measurements they cannot be used as a ground truth. Instead, we constructed a simple oscillator-based circuit to act as a gold-standard calibration target. It is based on a monostable configuration of a 555 timer and provides a controlled, repeatable delay: an input pulse is fed into the circuit, and it enables an output pin after a controllable time interval. That interval is set with a variable resistor, and in the current configuration it can be set between 1 and 500 ms. The delay was verified using an oscilloscope.

To test the Hammer, we connected the output of the oscillator to an LED with a ns-scale response time. We then connected the input of the oscillator to the signal from the Hammer that powers the relay. The Hammer's measurement head was placed so that it was sitting on top of the LED. No touchscreen device was used during these tests; instead, the LED played the role of the screen. When a Hammer measurement was run, the Hammer would power the relay to simulate a touch; the touch simulator would have no effect, since there was no touchscreen, but powering the relay would begin the oscillator's delay. After the specified time interval, the oscillator would turn on the LED, and the Hammer would record a response from the "screen". An oscilloscope was connected during each trial, and was used to measure the delay produced by circuit. We were then able to compare the latencies reported by the Hammer and by the oscilloscope. The results of our tests are shown in Figure 4, and demonstrate that the Hammer is repeatable within 0.4% across a range of latencies.

		Main Cluster			*Secondary Cluster*			
		Latency (ms)	SD (ms)	Cluster Size	Latency (ms)	SD (ms)	Cluster Size	Diff. from Baseline
iPad mini	Baseline	54.42	0.55	164	-	-	-	-
	+50 ms	103.7	1.09	158	120.3	0.91	34	49.3
	+100 ms	153.7	1.06	161	170	0.33	30	99.27
Nexus 9	Baseline	73.22	0.3	176	-	-	-	-
	+1 frame	89.92	0.41	130	103.3	1.35	36	16.7
	+2 frames	105.9	1.06	156	122.4	0.93	36	32.72

Table 1. Software Delay validation results. Each row shows one test condition containing 200 measurements. We show data for the two largest clusters (*Main* and *Secondary*) which constitute almost all measurements. The right column shows the difference between the baseline and the *Main* cluster, which matches the expected value. *Secondary* clusters are one 60 Hz display frame slower.

Camera-Based External Validation test

Though our Oscillator Test provided assurance of the absolute accuracy of our device, we wanted to perform an additional validation using a high-speed camera and an external method of triggering a touch. This orthogonal approach provided us with several benefits. First, it provided additional confidence that the Hammer was functioning as expected. Second, it provided a mechanism to cross-validate the Hammer's sensor inputs and ensure they are functioning correctly. Finally, we wished to conduct tests similar to those that have been done previously in the literature, such as by Ng et al. [15], in order to provide additional confidence.

The biggest challenge of camera-based validations of a tapping event is determining the moment of impact, which must be precisely measured in order to calculate latency. Most traditional capacitive styli have a solid internal tip (often made from brass) that is covered by a molded conductive rubber nub that is highly deformable. Unfortunately, this malleability means that the moment of screen contact can be somewhat ambiguous. When a stylus hits the screen, the initial point of impact will be the front edge of the rubber tip, which will begin to deform as the tip is compressed against the screen. The earliest stages of the impact may not trigger a touch event, and it can be difficult to visually determine the precise moment of impact. In order to remove this potential confound, we needed a stylus with a rigid tip. The tip must be conductive but still light and soft enough to avoid damaging the screen.

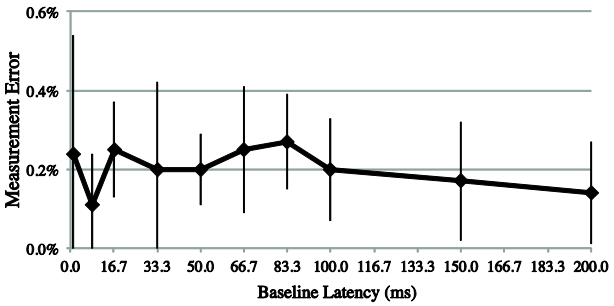

Figure 4. Oscillator validation results. Simultaneous measurements were taken with the Hammer and with an oscilloscope at a variety of latencies between 1 and 200 ms. All differences are below 0.4%, with most below 0.25%. Error bars show standard deviation.

A second complication with optical-based stylus measurements is that capacitive sensors can detect objects that are above the sensor, and not merely those that are in contact with it. Touch detection is based on thresholding a continuous signal that increases as an object gets closer to the sensor, and not on an inherently binary value. The thresholds are generally tuned so that the sensors detect input that is a few tenths of a mm above the surface. The problem with this detection height is that a device will receive a touch event when a stylus crosses the detection threshold, and not when it actually contacts with the screen itself. This means that a timing estimate based on the moment of contact may be slightly incorrect. Finally, because of the variance in latency on commodity devices, we need a way to measure the same touch event using both the Hammer and the high-speed camera, since we cannot directly compare sequential readings. To overcome these complications we designed the Direct Optical Hammer to externally validate the Latency Hammer, illustrated in Figure 5.

The Direct Optical Hammer is designed to produce a carefully controlled physical interaction with the screen, and allow that interaction to be easily timed using a consumer level high-speed but low-resolution camera. (We used a Nikon 1 J3, which can capture 1200 fps at a resolution of 320x120.) It also allows that interaction to be measured simultaneously by the Latency Hammer. The device is equipped with a custom capacitive stylus and extra apparatus to precisely record the moment of touch.

Custom Capacitive Stylus

The heart of the apparatus is a custom-built capacitive stylus. The body of the stylus is an aluminum rod (10 mm diameter), and the tip is a circular disk of UHMW polyethylene conductive plastic (12 mm diameter by 6 mm tall, sized to provide a contact area similar to a finger). This slippery plastic is extremely light and firm (similar to a plastic kitchen cutting board), and does not appreciably deform when it contacts the screen. The rod is held in place by a linear bearing which constrains the motion of the rod to a single axis. The bearing is attached to a custom 3D-printed mounting assembly that ensures the rod is perpendicular to the screen. The rod is connected with ultra-flexible wire (which has no impact on the movement of the rod) to a small circuit board in the mounting frame, and then to an electron sink similar to one used in the Hammer. This hardware configuration yields a capacitive stylus with a non-deformable tip that can be manually lifted a small distance off of a touchscreen, and then be precisely dropped onto the screen with no side-to-side movement.

This hardware is useful for conducting controlled observations, but it is not enough to allow an easy analysis via low-resolution high-speed footage.

Capturing Touch Time with a High-Speed Camera

To aid the analysis process, we added several additional components to create an onset-of-touch mechanical switch that is closed at the same instant the stylus triggers a touch event. The switch is constructed in two halves. The first is a 1 x 5 cm rigid cross piece added to the top of the rod (forming a "T"), with a brass contact plate on its bottom surface. The second half is a finely adjustable screw assembly mounted into the linear bearing parallel and immediately adjacent to the rod, with a brass plate on its top surface. The positioning of the screw and cross piece were arranged so that the contact plate on the cross piece will land on the contact on the screw. The screw acts as a stop that can prevent the stylus from travelling any further towards the screen, and the pair of brass plates will complete a circuit whenever the cross piece is resting on the screw. The utility of this apparatus stems from the choice of adjustment screw, which has extremely fine 100 TPI (threads per inch) threads. (As a comparison with a more common screw, the screw that connects a camera to a tripod is only 20 TPI.). The fine threads mean the height of the screw can be precisely controlled, since each full revolution of the screw changes its height by only 0.25 mm, and smaller adjustments can easily allow 0.01 mm changes.

Calibration and Operation of the Direct Optical Hammer

The Direct Optical Hammer must be calibrated so that the onset-of-touch switch triggers at the exact moment a touch is activated. First, the height of the screw is lowered so that it is below the level of the cross piece when the stylus is resting on the screen. Since the stylus is completely unimpeded by the screw, it will trigger a touch event, and will not close the onset-of-touch switch. We then raise the height of the screw until it comes into contact with the cross piece, which closes the switch but does not impact the touch event, since the stylus has not moved. We continue to raise the screw, which begins to lift the cross piece (and therefore the entire stylus) off of the screen. Eventually, the screw will lift the stylus far enough off the screen that it will cross the detection threshold, and a touch event will no longer be triggered. At this point we lower the screw until a touch event occurs. After a series of minute adjustments (which include lifting and dropping the stylus several times after each adjustment), we can calibrate the height of the screw so that the stylus will be stopped (and the onset-of-touch signal generated) at the exact height that triggers a touch event. We connect this

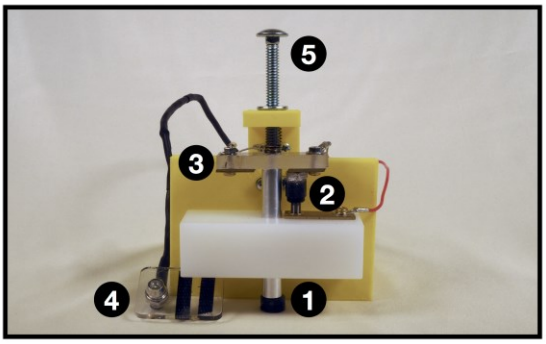

Figure 5. The Direct Optical Hammer is built around a custom capacitive stylus (1). An onset-of-touch signal is created when the adjustable screw (2) makes contact with a cross piece (3), and illuminates the LED (4). The speed of the stylus drop is controlled by an upper stop (5) that ensures the stylus is always dropped from the same height.

signal to an LED with a ns-scale response time to provide an easily discernible visual signal on a low resolution video. Finally, we added an upper stop so that the stylus was always dropped from the same height to ensure the same downward velocity, and a spring to prevent the stylus from bouncing when it hits the screen. With this calibration complete, we can assemble the experimental setup shown in Figure 5.

Validation

A Nexus 9 tablet was placed under the Latency Hammer, and our high-speed camera was positioned so that it was pointing down at the tablet. The Direct Optical Hammer was placed on the tablet, and the LED positioned so that it was clearly visible in the camera image. The onset-of-touch signal was connected to the relay closure detection circuit in the Latency Hammer, and the power to the relay was disconnected. This meant that nothing would happen when the Hammer tried to activate the relay to simulate a touch, but the Hammer would nonetheless detect a signal putatively indicating that the relay had closed. In other words, from the Hammer's point of view, a calibration trial was identical to a regular trial except that it took the "relay" several seconds to close (i.e., the time required to lift and drop the stylus), rather than the couple of ms that an actual relay would require. Since the Hammer automatically subtracts the relay closing time from its measured latency, we generated valid readings without requiring any hardware or software change, which enabled us to test the Hammer in a configuration that was as close to normal as possible. The Hammer measurement head was positioned so that it was aligned with the stylus, and an oscilloscope was connected to the onset-of-touch signal to determine if the stylus bounced when it hit the screen. Bounces were extremely small and not visible to the naked eye, but could occur due to the mechanical forces involved. A bounce-free impact results in a step function in the onset-of-touch signal: it goes from low to high in a single transition. A bounce manifests itself as a series of low-high-low-high transitions, which usually occurred over a period of a few ms. Because bounces would add an additional confounding variable (i.e., it would be unclear which impact the device responding to), trials that bounced were discarded.

A trial consisted of the following steps. The Hammer was commanded to begin a reading; no relay activity (and therefore no simulated touch) would occur, since the relay's power had been disconnected. The high-speed camera was started, and the stylus was then manually lifted to the upper stop and gently released, imparting as little vertical velocity as possible. At the moment that the touchscreen registered a stouch event from the falling stylus, the onset-of-touch signal illuminated the LED and told the Hammer that its relay had closed. The Hammer would begin monitoring its photodiode, and then report a latency. The high-speed footage was then analyzed manually, and the frame numbers corresponding to the LED illumination and screen response were recorded. We calculated the latency based on the high-speed footage and compared it to the latency reported by the Hammer.

Results

The optical latency measurements have an uncertainty corresponding to two camera frames (1.66 ms) due to resolution and frame rate limitations. Like most screens, the LCD in the Nexus 9 refreshes from top to bottom. When viewed at 1200 fps, this refresh is clearly visible, and the screen's transition from black to white as it redraws in response to the touch event progresses at approximately 8 mm per camera frame. Low resolution footage of this redraw progression makes it somewhat ambiguous as to the exact moment when the photodiode will be sufficiently illuminated to generate a "screen is on" signal, although it is clear that it happens within the two frame window.

We conducted a series of 20 combined samples with a Nexus 9 under normal conditions (i.e., radio on, random background applications running on the device.) The latency measured by each apparatus varied from 75 to 200 ms across the 20 samples. The average difference for a given sample between the optical and Hammer latency measurements was 0.88% (SD = 0.39%). Due to the footage uncertainties discussed above, we also report the difference for the frame where the display has redrawn beyond the photodiode (i.e., the worst case scenario); this resulted in an average difference of 1.6% (SD = 0.59%).

These validation techniques all demonstrate the accuracy and reliability of the Hammer. However, measuring the latency of real-world commodity devices can be complex, and we now report on some of these challenges.

MEASURING LATENCY IN THE REAL-WORLD

During the development of our latency measurement tools and the analysis of many different devices, we have found that latency can be a remarkably fuzzy concept. In general, when dealing with a commodity device running a standard operating system, it does not make sense to speak of "the latency" of the device. Instead, we routinely see the same device exhibiting a wide range of latencies, sometimes varying by as many as 3 frames (50 ms) between sequential samples, as exemplified in Figure 6. This phenomenon may not occur on specialized devices such as the demonstrator systems described previously in the literature [11, 15], but it is evident and widespread on commercially-available devices.

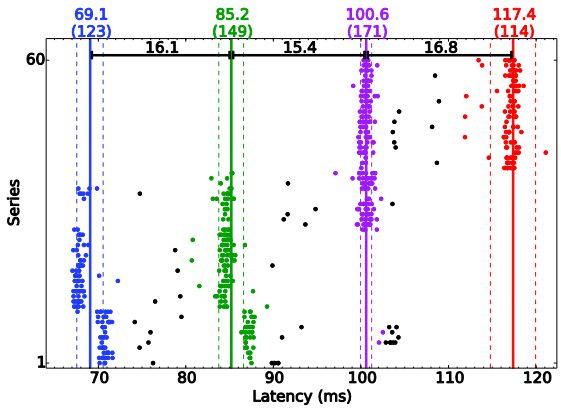

Figure 6. Raw results of 600 latency measurements on a Nexus 5 running Android 5.1.1. The devices exhibits a latency profile that changes over time.

These variations in performance are not surprising, given the complex nature of the systems in question. Some of the factors that we have observed impacting performance are:

Heat. On many systems, as their internal temperature increases, the OS may choose to disable or throttle one or more CPU cores to reduce the thermal load. This effect is more pronounced on small devices (e.g., phones or tablets), since they often lack fans or other active thermal management technologies and thus have fewer options to cool themselves. We observed the effect of thermal throttling during longer measurement runs on some devices. For example, on the dataset from a Nexus 5 phone shown in Figure 6, we found that our initial sets of measurements were split between clusters at 69.1 and 85.2 ms. However, after approximately 300 measurements, an additional frame of latency would suddenly appear, and we would record latency clusters at 100.6 and 117.4 ms. Allowing the phone to cool off (or briefly placing it in a freezer) would restore the faster performance.

System Load. As we would expect, a busy OS handling a large number of processes will perform worse than a system that is idle. Reducing the number of background tasks (e.g., quitting other apps, disabling radios via Airplane mode, etc.) would often improve device latency.

Battery State. Similar to heat, throttling may occur if the device's battery level is low, and the OS switches off or throttles CPU cores in an attempt to save power.

Vertical Screen Refresh. As discussed above in the section on the Direct Optical Hammer, most screens redraw from top to bottom. This redraw time is small, but can be on the order of ms. For example, we observed that a Nexus 9 could take upwards of 10 ms to redraw its screen. This means that the location of the measurement head can have an impact, since the same redraw operation will appear to take less time if the screen response is measured at the top rather than the bottom of the screen. The position of the measurement head must therefore be standardized (e.g., at the center of the screen). In this paper, all measurements were obtained from the center of the screen, a position we assume is centered in the redraw step.

Some of these issues do not have clear answers, since it depends on the purpose of the latency measurement. In other words, when we measure the latency of a device, are we interested in best possible performance that the hardware can exhibit (e.g., a cold, idle phone in Airplane mode), or the worst performance that a user is likely to encounter in the wild (e.g., a hot phone with a low battery, running many background tasks while connected to a busy Wi-Fi network)? The answer will depend on the context of the measurements. In either case, it is important to think of latency as a range of possible values, rather than a single number. However, all of these variables can be quantified, and can often be obtained programmatically by the measurement applications (e.g., querying the OS to determine the current power saving mode), thus adding context to the obtained measurements. Furthermore, the automation facilitated by the Hammer allows us to consider latency within the context of the complexity of real-world devices in real-world conditions, which has not been possible with previous latency measurement devices explored by the community.

Sample Data

We now present some latency measurements for recent commercial devices, which are shown in Table 2. As in Table 1, we present the two largest clusters of measurements. 500 measurements were taken for each device. As we have discussed, many factors can impact the latency of a device. While we have attempted to control for many of these (e.g., low temperature, low load, full battery), others, such as differences in software, are beyond our control. As such, it is important to understand that this data, while accurate and representative, does not lend itself to cross-device comparisons. Just as fuel efficiency standards provide multiple ratings ("city" and "highway"), there is a clear need to develop a framework for the conditions under which latencies are measured with a device such as the Hammer. This is an important area of future work.

FUTURE WORK AND CONCLUSION

We have presented the Latency Hammer, a tool that empirically measures the interface latency of touchscreen devices. The Hammer is low-cost and capable of measuring latency with little or no human involvement. We also provide software tools that automatically analyze the resulting data.

We have also presented a series of hardware and software tools used to evaluate the Hammer, and demonstrated that the Hammer is repeatable to within 0.4% between successive measurements of a simplified oscillator circuit and that its measurements are within 0.8%–1.6% of measurements taken with a high-speed camera. These validation tools can be used to benchmark other latency measurement equipment.

We intend to continue development of the Latency Hammer system and to further improve the robustness of the design (e.g., moving the ADC into the measurement head to reduce noise). We are keenly interested in adapting the Hammer (including some of the approaches used in the Direct Optical Hammer) to test dragging performance, the impact of multitouch, and to test non-capacitive sensors (e.g., EMR pens, or optical sensors). Finally, we intend to continue the development of our benchmarking tools and validation techniques in order to provide further confidence in the robustness of our systems and to provide a standard set of benchmarks for others.

The Hammer is a powerful tool that can be used by system builders at both the hardware and software level. We hope that the low-cost and availability of such a measurement tool will speed and simplify system development and improvement.

ACKNOWLEDGEMENTS

Thanks to our colleagues at Tactual Labs and at the DGP lab at the University of Toronto. Special thanks to Michael Glueck, Katie Barker, John Hancock, and Preksha Kashyap.

	Main Cluster			*Secondary Cluster*		
Device	Latency (ms)	SD (ms)	Cluster Size	Latency (ms)	SD (ms)	Cluster Size
Nexus 6 (Android 5.1.1)	92.35	1.28	392	78.33	1.5	82
iPhone 6 (iOS 8.4.1)	52.35	1.33	455	-	-	-
Nexus 9 (Android 5.1.1)	87.19	0.77	253	70.66	1.27	226
Surface Pro 3 (Windows 10)	69.53	1.76	354	101.7	1.07	70

Table 2. Results of 500 measurements for several commercial devices. The two largest clusters are reported.

REFERENCES

1. Glen Anderson, Rita Doherty, and Subhashini Ganapathy. 2011. User Perception of Touch Screen Latency. In *Proceedings of Design, User Experience, and Usability (DUXU '11)*. Springer-Verlag GmbH Berlin Heidelberg, 195–202. DOI: http://dx.doi.org/10.1007/978-3-642-21675-6_23
2. Michelle Annett, Albert Ng, Paul Dietz, Walter F. Bischof, and Anoop Gupta. 2014. How Low Should We Go? Understanding the Perception of Latency While Inking. In *Proceedings of Graphics Interface 2014 (GI '14)*. Canadian Information Processing Society, Toronto, ON, Canada, 167–174. http://dl.acm.org/citation.cfm?id= 2619648.2619677
3. Arduino LLC. 2012. Arduino Due microcontroller board. Retrieved January 8, 2016 from https://www.arduino.cc/en/Main/ArduinoBoardDue.
4. François Bérard and Renaud Blanch. 2013. Two Touch System Latency Estimators: High Accuracy and Low Overhead. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces (ITS '13)*. ACM, New York, NY, USA, 241–250. DOI: http://dx.doi.org/10.1145/2512349.2512796
5. Géry Casiez, Stéphane Conversy, Matthieu Falce, Stéphane Huot, and Nicolas Roussel. 2015. Looking through the Eye of the Mouse: A Simple Method for Measuring End-to-end Latency using an Optical Mouse. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology (UIST '15)*. ACM, New York, NY, USA, 629–636. DOI: http://dx.doi.org/10.1145/2807442.2807454
6. Elie Cattan and Francois Bérard. 2015. A Predictive Approach for an End-to-End Touch-Latency Measurement. In *Proceedings of the 2015 ACM International Conference on Interactive Tabletops and Surfaces (ITS '15)*. ACM, New York, NY, USA, 215–218. DOI: http://dx.doi.org/10.1145/2817721.2817747
7. Jonathan Deber, Ricardo Jota, Clifton Forlines, and Daniel Wigdor. 2015. How Much Faster is Fast Enough? User Perception of Latency & Latency Improvements in Direct and Indirect Touch. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1827–1836. DOI: http://dx.doi.org/10.1145/2702123.2702300
8. Ricardo Jota, Albert Ng, Paul Dietz, and Daniel Wigdor. 2013. How Fast is Fast Enough? A Study of the Effects of Latency in Direct-touch Pointing Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2291–2300. DOI: http://dx.doi.org/10.1145/2470654.2481317
9. Topi Kaaresoja and Stephen Brewster. 2010. Feedback is... Late: Measuring Multimodal Delays in Mobile Device Touchscreen Interaction. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction (ICMI-MLMI '10)*. ACM, New York, NY, USA, Article 2, 8 pages. DOI: http://dx.doi.org/10.1145/1891903.1891907
10. Joseph J. LaViola. 2003. Double Exponential Smoothing: An Alternative to Kalman Filter-based Predictive Tracking. In *Proceedings of the Workshop on Virtual Environments 2003 (EGVE'03)*. ACM, New York, NY, USA, 199–206. DOI: http://dx.doi.org/10.1145/769953.769976
11. Darren Leigh, Clifton Forlines, Ricardo Jota, Steven Sanders, and Daniel Wigdor. 2014. High Rate, Low-latency Multi-touch Sensing with Simultaneous Orthogonal Multiplexing. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 355–364. DOI: http://dx.doi.org/10.1145/2642918.2647353
12. Jiandong Liang, Chris Shaw, and Mark Green. 1991. On Temporal-spatial Realism in the Virtual Reality Environment. In *Proceedings of the 4th Annual ACM Symposium on User Interface Software and Technology (UIST '91)*. ACM, New York, NY, USA, 19–25. DOI: http://dx.doi.org/10.1145/120782.120784
13. Robert B. Miller. 1968. Response Time in Man-computer Conversational Transactions. In *Proceedings of the December 9–11, 1968, Fall Joint Computer Conference, Part I (AFIPS '68 (Fall, part I))*. ACM, New York, NY, USA, 267–277. DOI: http://dx.doi.org/10.1145/1476589.1476628
14. Albert Ng, Michelle Annett, Paul Dietz, Anoop Gupta, and Walter F. Bischof. 2014. In the Blink of an Eye: Investigating Latency Perception During Stylus Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1103–1112. DOI: http://dx.doi.org/10.1145/2556288.2557037
15. Albert Ng, Julian Lepinski, Daniel Wigdor, Steven Sanders, and Paul Dietz. 2012. Designing for Low-latency Direct-touch Input. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 453–464. DOI: http://dx.doi.org/10.1145/2380116.2380174
16. Ohmite Manufacturing Company. 2011. R2V-CT6-38E Heatsink. Retrieved January 8, 2016 from http://www.ohmite.com/cat/sink_r2.pdf.

17. Omron Corporation. 2014. G6J-2P-Y Ultra-Small and Slim DPDT Relay. Retrieved January 8, 2016 from http://www.omron.com/ecb/products/pdf/en-g6j_y.pdf.
18. OptoFidelity LLC. 2015. OptoFidelity WatchDog. Retrieved January 8, 2016 from http://www.optofidelity.com.
19. Andriy Pavlovych and Wolfgang Stuerzlinger. 2009. The Tradeoff Between Spatial Jitter and Latency in Pointing Tasks. In *Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '09)*. ACM, New York, NY, USA, 187–196. DOI: http://dx.doi.org/10.1145/1570433.1570469
20. Vishay Semiconductors. 2014. BPW46 Silicon PIN Photodiode. Retrieved January 8, 2016 from http://www.vishay.com/docs/81524/bpw46.pdf.
21. Anthony Steed. 2008. A Simple Method for Estimating the Latency of Interactive, Real-time Graphics Simulations. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology (VRST '08)*. ACM, New York, NY, USA, 123–129. DOI: http://dx.doi.org/10.1145/1450579.1450606
22. Colin Swindells, John C. Dill, and Kellogg S. Booth. 2000. System Lag Tests for Augmented and Virtual Environments. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST '00)*. ACM, New York, NY, USA, 161–170. DOI: http://dx.doi.org/10.1145/354401.354444
23. William Thomson 1st Baron Kelvin. 1891. Electrical Units of Measurement. In *Popular Lectures and Addresses Vol 1: Constitution of Matter*. MacMillan and Co, London, UK, 80–81.
24. Colin Ware and Ravin Balakrishnan. 1994. Reaching for Objects in VR Displays: Lag and Frame Rate. *ACM Transactions on Computer-Human Interaction (TOCHI)* 1, 4 (Dec. 1994), 331–356. DOI: http://dx.doi.org/10.1145/198425.198426
25. Neng-Hao Yu, Li-Wei Chan, Seng Yong Lau, Sung-Sheng Tsai, I-Chun Hsiao, Dian-Je Tsai, Fang-I Hsiao, Lung-Pan Cheng, Mike Chen, Polly Huang, and Yi-Ping Hung. 2011. TUIC: Enabling Tangible Interaction on Capacitive Multi-touch Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2995–3004. DOI: http://dx.doi.org/10.1145/1978942.1979386