Two Touch System Latency Estimators: High Accuracy and Low Overhead

François Bérard, Renaud Blanch
LIG, University of Grenoble, Grenoble-INP, UJF
BP 53, 38041 Grenoble Cedex 9, FRANCE
{francois.berard, renaud.blanch}@imag.fr
+33-476-514-365

ABSTRACT

The end-to-end latency of interactive systems is well known to degrade user's performance. Touch systems exhibit notable amount of latencies, but it is seldom characterized, probably because latency estimation is a difficult and time consuming undertaking. In this paper, we introduce two novel approaches to estimate the latency of touch systems. Both approaches require an operator to slide a finger on the touch surface, and provide automatic processing of the recorded data. The High Accuracy (HA) approach requires an external camera and careful calibration, but provides a large sample set of accurate latency estimations. The Low Overhead (LO) approach, while not offering as much accuracy as the HA approach, does not require any additional equipment and is implemented in a few lines of code. In a set of experiments, we show that the HA approach can generate a highly detailed picture of the latency distribution of the system, and that the LO approach provides average latency estimates no further than 4 ms from the HA estimate.

Author Keywords

Latency, Touch.

ACM Classification Keywords

H.5.2 User Interfaces: Input devices and strategies (e.g., mouse, touchscreen)

General Terms

Human Factors, measurement, performance.

INTRODUCTION

Every interactive system exhibits some amount of end-to-end latency: the delay between an action of the user on the input device, and the display of the corresponding feedback. Latency has a strong negative effect on user's performance (both task accomplishment time and error rate) even at levels as low as 100 ms [8]. Latency on direct-touch system is easy to perceive: even a small temporal gap results in a clearly visible spatial offset between the finger and the virtual object controlled by the finger. The offset remains visible as long as the finger is moving. Users can perceive latency on a direct touch surface at levels lower than 3 ms [10]. A more recent study seems to indicate that latency on direct touch devices affects user's performance in pointing tasks even at a level as low as 1 ms [6]. Current commercial direct-touch devices have latencies estimated in the range 50 ms to 200 ms [10]. These devices are thus far from optimal with respect to user's capabilities. Moreover, higher than optimal latencies are not restricted to commercial devices, they are frequent in experimental prototypes. This is particularly an issue when the degradation of user's performance due to latency can mask the benefits of a new interaction technique implemented on the prototype.

Despite its strong effect on user's performance, the problem of latency is not widely acknowledged and evaluated. Consumer touch devices are typically rated for their screen size and definition, but their technical specifications never includes latency. In research, even recent studies on touch interaction, such as modeling target acquisition [2], do not estimate nor mention latency. This may be the consequence of the difficulty to accurately estimate interactive systems latency. The literature offers several approaches based on the recording of a video of both the device and the graphical output, and then analyzing the video image by image to measure a gap between the device and the output ([7, 10, 11, 13, 14, 15]). This process is time consuming and often requires a complex physical setup to move the device in a controlled manner. Furthermore, the manual aspect of the process limits the number of latency estimations that can be performed. But we will show that a statistical estimation (i.e. relying on many repeated measures) is often a requirement for an accurate estimation of latency.

In this paper, after detailing the difficulties of latency estimation, and reviewing previous works, we introduce a novel "High Accuracy" (HA) approach to touch device latency estimation. It requires an external camera and careful calibration, but the data processing is automatic and results in a large sample set of accurate latency estimations. We then introduce a second "Low Overhead" (LO) approach which, while not offering as much accuracy as the HA approach, does not require any additional equipment in addition to the touch device, and can be implemented in a few lines of code. We run a set of experiments to compare latency estimates based on the HA ap-

ITS'13, October 6–9, 2013, St. Andrews, United Kingdom.

ACM 978-1-4503-2271-3/13/10...$15.00.
http://dx.doi.org/10.1145/2512349.2512796

proach, the LO approach, and a high speed recording (HSR) approach from the literature. Finally, we discuss the results of the experiments and conclude.

LATENCY ESTIMATION

The latency of an interactive system is the delay between a user's action and the time the corresponding feedback is presented to the user. There can be many sources of latency, the most common sources being the input device sensing time, the communication time between the input device and the computer, and the time to generate and display the feedback. In this section, we present the basic principles of latency estimation in the general case, i.e. principles that apply to both direct (e.g. touch) and indirect (e.g. mouse) devices.

Regardless of the sources of end-to-end latency, it has been measured either by observing a *spatial* gap or a *temporal* gap between a moving input device and the corresponding visual feedback. In both approaches, the gaps are estimated from pictures recorded with a camera and showing both the device and the graphical feedback moving along some trajectory[1].

The measure of a *temporal* gap provides a direct estimation of latency. This can be done by choosing a landmark in the video and counting the number of images between the physical device and the graphical feedback reaching the landmark. The landmark does not have to be at the same location for the device and feedback trajectories in the recorded images: it can be a characteristic event such as reaching an extremum. The main limitation of this "image counting" estimation is that its precision is bounded by the frequency of the camera. Even a high speed video camera at 250 images/s only provides a precision of 4 ms. Another limitation of this approach is that it discards most of the recorded images, using only the ones where the device and the feedback reach the landmark.

Latency can also be estimated by observing a *spatial* gap of length g, in the recorded pictures, between the device and the feedback. This is illustrated on Figure 1 in the case of a direct-touch device. This approach requires the estimation of the speed of the device s at the time the image was shot, in addition to the estimation of the gap length. Latency is then computed as g/s. The precision of this "speed" approach is thus only bounded by the precision of the gap and speed estimations, it does not have the hard limit of the camera frequency as with the previous method. In addition, all the images in the recorded video can be processed to provide a latency estimate. This approach, however, requires to superimpose the trajectories of the device and the graphical feedback in the recorded images. While this is a natural feature of direct-touch devices, it usually requires an ad-hoc physical setup for indirect devices such as a computer mouse.

Setting aside the approach used, a single latency estimate is not enough to represent the complex evolution of latency in time. Users are exposed to the system latency in a continuous world on a display that is updated in a discrete way, typically at 60 Hz. When the display has just been refreshed, users are exposed to a *minimal* gap, corresponding to a minimal latency l_{min}. But, until the next refresh, the feedback can not be updated while the finger carries on its continuous motion. Hence users are exposed to a continuously increasing gap, and increasing latency, until just before the next refresh when the gap and latency l_{max} are *maximal*. One way to deal with this phenomenon is to attempt to find l_{min} and l_{max} in the recorded images, and to consider that on average users are shown a latency of $(l_{min}+l_{max})/2$. We call this the "min-max" approach.

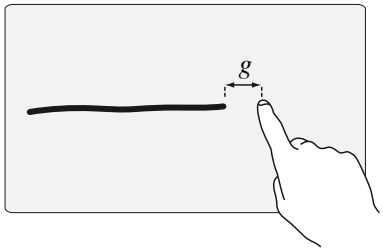

Figure 1. Principles of latency estimation in the case of a direct touch device, after [10]. A picture is shot showing both the finger and the feedback (the bold stroke). The length of the gap g between the two is measured. Assuming an estimated speed of the finger s, latency is g/s.

Estimating l_{min} and l_{max}, however, is viable only if the system exhibits a constant latency for all its components up to the display. This is often *not* the case: the different components of the system generate variable amounts of latency. The rendering time of the graphical feedback, for example, can vary significantly depending on the complexity of the update. Even a small increase of the rendering time may make it pass the display refresh period, resulting in the end-to-end latency augmented by a whole refresh period. In addition, the interaction between the components themselves is a source of variability: in general, input devices are not synchronized with the display. An input device may sample at 100 Hz (a common case for a computer mouse) when the display refreshes at 60 Hz. In the best case, the display is refreshed just after the input device has produced a new sample. But in the worse case, the input device's sample may be close to 10 ms old, adding as much to the end-to-end latency.

Because of the complex evolution of latency over time, the average $(l_{min}+l_{max})/2$ is not an accurate description of the latency distribution. Moreover, the occurrence of the true l_{min} and l_{max} (i.e. accounting for all sources of variation) may be low, making impracticable their search "by hand" in the recorded video. It appears that a high number of observed latencies are required to model its distribution, which calls for an automated approach.

RELATED WORK

Efforts to estimate device's latency have been historically focused on the 3D tracking required in virtual reality systems, as these systems tended to present high levels of latency that hindered the feeling of immersion [1, 7, 9, 12, 13, 14, 15, 16].

The work by Liang et al. is the first report of a latency estimator based on the recording of a video of the physical device and the system's graphical feedback [7]. The physical device

[1] In the paper, we call the pictures recorded by the camera *camera images* or *images*, and the pictures displayed as visual feedback *display frames* or *frames*.

was attached to a pendulum so that its motion was known. Latency was computed from the estimated spatial gap and pendulum speed, using the "speed" approach. The pendulum trajectory was used in several subsequent efforts [9, 14, 16]. Ware et al. also used the "speed" approach but with a stepper motor moving the device back and forth in a linear motion, instead of a pendulum [15]. Swindells et al. used the "speed" approach with a turntable moving the input device in a circular motion at constant speed [13]. In all these studies, the analysis of the recorded pictures was manual, preventing the collection of a large statistical sample of latencies. In addition, these studies relied on the input device moving along a perfectly known trajectory so that its speed was known with great accuracy. This approach is impracticable in the case of many direct-touch input devices (e.g. capacitive) that require an actual human finger for detection.

Pavlovych et al. did not rely on a well controlled trajectory: a computer mouse was moved by hand back and forth along the top edge of a display bezel [11]. Latency was estimated as the *temporal* gap between the mouse and the cursor reaching the edge of the display using the "image counting" approach. The estimation was made at the resolution of the camera period: 16.7 ms for a 60 Hz recording. Analyzing 2 minutes of video to make 10 measurements and then averaging them improved the precision of the estimation.

Mine used a pendulum as in Liang et al., but introduced the use of photodiodes to automatically detect the time when the pendulum reached its vertical position and the time when the update was displayed on a monitor [9]. This is the first reported work that does not require the video recording of the device and its subsequent analysis. The technique, however, requires a specialized and expensive piece of equipment: a recording oscilloscope was used to observe the timings of the signals from the photodiodes. In addition, the analysis of the signals remained manual.

To our knowledge, Steed's approach is the only one allowing the automatic estimation of latency by a system [12]. As in Liang et al., the motion of a physical pendulum and its graphical feedback was recorded in a video, but Steed's approach did not require the two trajectories to be aligned. The background of the scene was set to black, and a small light was attached to the input device. This allowed the system to automatically extract from the recorded images both the position of the device and the position of the graphical feedback. Two sine functions were then fitted to the extracted trajectories, and the latency was estimated as the phase shift between the two sines. The trajectory having the shape of a sine makes this approach difficult to apply to direct touch. Moreover, the function fitting step averages the variations of latency into a single fit, producing a single latency estimate.

All the approaches described above were designed for systems using indirect devices. Measuring the latency of direct-touch systems is a much more recent concern. Ng et al. offer the first report of a method based on the "speed" and "min-max" approaches [10]. This method is illustrated on Figure 1 and Figure 7. A high frequency video camera (240 Hz) records a finger sliding along a graduated ruler on a touch surface. The feedback is a simple line showing the finger's trajectory. The speed of the finger is estimated by using the ruler to measure the distance travelled by the finger between two chosen camera images, and using the camera image period to estimate the time elapsed between these two images. The speed estimate is then used in individual images to convert the spatial gap between the finger and the graphical feedback into latency. Looking at the recorded video image by image, the authors find the images with the minimal and maximal gap, divide the gaps by the speed to estimate the minimal and maximal latency, and average the two as an estimate of the average latency. As explained in the previous section, this min/max approach only works because the touch surface used by Ng et al. is custom build and synchronizes the sensing with the display. As a result, the only source of latency variation is the display frame period. Another limitation of this approach is that, in order to estimate the gap between the graphical feedback and the finger, the touch position must be inferred from a top view of the finger. This is likely to introduce some inaccuracies in the estimation. We will illustrate this problem in the "Experiments" section.

HIGH ACCURACY TOUCH LATENCY ESTIMATION

In order to achieve high accuracy in touch latency estimation, we design a new approach that builds on and improves upon previous approaches. Its principles are illustrated on Figure 2. As with previous approaches, we use an external camera to capture frozen images of the physical position of the device (the finger) and the current state of the system (the number of the frame currently being displayed). We estimate latency as the time difference between the date of the frame currently displayed, and the date the recorded trajectory will reach the current touch position of the finger. The current touch position is computed from a clearly visible pattern attached to the finger, as illustrated on Figure 3. Our approach thus provides a direct estimate of latency (i.e. it does not require an estimation of the speed of the finger). This approach has three main benefits compared to the literature:

- we use the recorded trajectory as our time measurement instrument. This removes the precision bottleneck of the

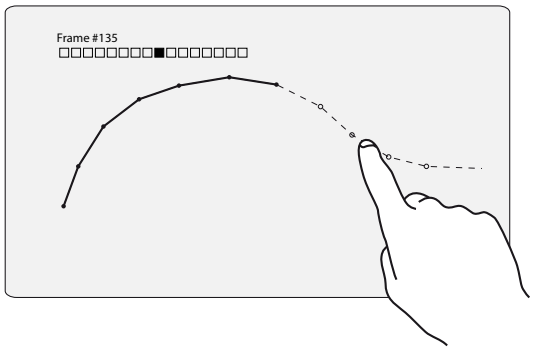

Figure 2. Principles of the High Accuracy (HA) estimation. The displayed frame counter (#135) is encoded by the single displayed bit (black square). The plain thick line represents the *event trajectory*: the events received by the system so far. The dashed line is the *future trajectory*. Latency is the time it will take for the *event trajectory* to reach the current physical finger position.

camera image counting approach used in previous direct estimations of latency.

- we use a visual pattern attached to the finger to allow the extraction of an accurate touch position from images of the finger as viewed from above.
- we use simple visual patterns to encode the number of the current display frame and to locate the finger touch position. This allows automated processing and thus the collection of a large sample set of latencies.

More specifically, our approach is made of the following steps:

- Capture an image of the finger and the display showing the current frame number.
- Process the captured image to extract the current frame number and the position of the physical finger.
- Perform a lookup in the frame log to find the display date d_i of the current frame. This is also the date when the camera image was captured.
- Perform a lookup in the event log to find the date d_e when the trajectory reaches the observed physical finger position.
- Estimate the latency as $d_e - d_i$.

The event trajectory is only shown on Figure 2 for illustration purpose: our approach does not need to perform any image processing with it, and thus it is not displayed by the system. Also, at the time the image is acquired by the camera, the event log obviously does not contain its future. Hence latency can not be estimated on the fly: camera images, frame log and event log are recorded for short sessions (a few seconds), and latency is computed on this data after the recording.

These simple principles present a number of difficulties: converting coordinates back and forth the display and camera coordinate systems, extracting the touch position from an image of a finger, extracting the frame counter, and finally finding the date when the trajectory reaches the observed touch position. We detail these difficulties in the following sections and present our solutions.

Camera-Display Calibration

Our approach requires that the *observed touch position* recovered from the captured image is searched along the system event trajectory expressed in display coordinates. We thus need a fairly accurate correspondence between the camera and the display coordinate systems.

We assume a pure perspective projection between the display plane and the camera projection plane, i.e. a homography. We calibrate the homography by recording an image of a black display, and then 12 successive images, each capturing a single dot displayed using a regular grid at known display coordinates. We compute the difference image between the black image and each dot images, find the biggest set of connected pixels in the difference image, and compute the center of the set. This center is used as the dot's image coordinates. From this set of 12 correspondences, we perform a least-square estimation of the homography matrix. Using this matrix H and its inverse H^{-1}, we can convert coordinates from display to camera, and from camera to display.

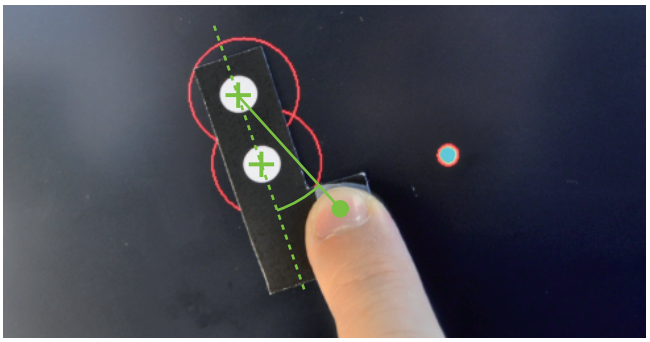

Figure 3. Result of the camera-touch calibration. The green annotations have been added to the picture. The paper pattern (white dots on a black background) is attached to the finger. The large red circles indicate the detected position of the pattern's dots. The small red disk shows the result of the transformation (d,θ) applied to the dots, but with an offset to move it out of finger occlusion. The small green disk shows the currently measured touch position, with the same offset.

Camera-Touch Calibration

Our approach requires that we estimate touch positions from camera images of a finger as seen from above. However, when touching a display, a finger hides the touch position. Furthermore, visually tracking fingers, which are non-rigid objects, is a difficult process that may not yield enough accuracy for our purpose. We solve both problems by printing a small pattern on a piece of paper. The pattern is about $5\,\text{cm} \times 2.5\,\text{cm}$, it is made of 2 large white dots (8 mm radius) on a black background. We attach this pattern to the touching finger by means of double sided tape. The thin sheet of paper does not hinder the capacitive touch sensor, but it provides an easily detectable pattern at a fixed transformation (translation and rotation) from the touch location, as illustrated on Figure 3.

We calibrate the transformation between the pattern and the touch position by recording the touch events and camera images for a small time span (1 s) during which the finger must *remain stable*. The procedure is cancelled if the finger moves by more than a pixel. At the end of the recording, we locate the two dots of the pattern in the recorded images (about 50 images are recorded in 1 s). The dots appear as the brightest objects in the scene and can be easily located: we simply threshold the image, perform connected pixels analysis, and filter the resulting connected pixel groups according to their shape (expected size and aspect ratio close to 1.0). The center of the two sets of connected pixels is used as the dots' coordinates in the image. We transform these two centers into display coordinates using the inverse homography matrix H^{-1}.

We then compute the average dot positions and touch event position across the 1 s recording. We use the most distant dot from the touch location as the origin of the transformation, and define the transformation as $T(d,\theta)$ where d is the distance between the origin and the touch position, and θ is the

angle between the dots line and the line passing through the origin and the touch position. This transformation is illustrated on Figure 3. We repeat this procedure at different location of the display and different finger orientation and verify that the calibrated transformation remains stable.

Dating the captured image

Our latency estimation requires that we get a fairly accurate estimation of the capture date d_i of the image. It is not possible to rely on the date when the image is received by the system: it may include several delays due to the camera itself, the transmission of the image over the IEEE1394 link, and the IEEE1394 driver stack on the computer. We thus use a visual tag to synchronize the camera acquisition with the display.

We increment a frame counter at the display frequency (i.e. at 60 Hz). The frame counter is represented on each displayed frame by both a number and a tag. The tag is chosen so that it can be recovered from a camera image by simple image processing, it is illustrated as the bit vector at the top of Figure 2. We use a white square of 15×15 pixels to represent an "on" bit; "off" bits are not displayed. As we know where we draw the bits on the display, we can use the homography matrix H to transform bit positions into image coordinates. We thus know where to look for each bit in the captured image.

We initially displayed a direct binary representation of the frame counter in the bit vector. However, sometimes the camera grabs an image of the display while it is switching to the next frames. Bits of the previous frame are being replaced by bits of the next one. This results is some *grey* bits which can be classified either as "on" or "off", resulting in an erroneous value for the frame counter. We thus resorted to a simpler bit pattern. We display 16 bits, only one of them is "on" at any time: the "on" bit represents the remainder of the integer division of the frame counter by 16. As there is only a single "on" bit in every image, the processing becomes a simple search for the brightest bit. This is robust to the display's frame switch: only very dark images, exactly at the frame refresh, will not yield a frame counter, but there is no risk to misinterpret the bits as with the binary representation. However, this frame counter representation is incomplete: it does not allow to extract the frame counter from a single image. We simply increment the frame counter by 16 when we observe that this remainder drops from one image to the next one.

This process allows us to robustly associate a camera image to the display frame it captured. To get the *date* d_i of the image capture, we maintain a log file of the dates of the frame displays. We first planned to implement this log file by storing the system's clock after the "swap_buffer" call in a standard double-buffer display loop synced with the display refresh. Using GLUT/OpenGL for the rendering, this was more complex than anticipated: OpenGL being optimized for throughput, not latency, the glutSwapBuffer call is pipelined and returns without blocking before the frame is actually shown on the display. We use a solution found in the implementation of the Psychophysics Toolbox [3]: after the "swap_buffer" call, we draw some pixels on the new back-buffer and require OpenGL to synchronize with a call to glFinish(). We verified that this effectively makes glFinish() to block until the first buffer is shown on the display. We can then use the system's clock at the return of glFinish() as the date of the frame.

Another difficulty is that each display frame is shown for the duration of the refresh period: 16.7 ms on our 60 Hz display. The system clock at return of glFinish() corresponds to be the beginning of this period. We have no way, however, to estimate at which point of this 16.7 ms refresh period the image was captured. This is not a problem for our purpose, as it provides a way to measure various samples in the best case/worst case range. Our camera is not synchronized with the display and acquires images at a different frequency. We plot a histogram of the offsets produced by the combination of the camera's 49.7 Hz capture rate and the display's 60 Hz refresh rate. The histogram shows that the sampling of the refresh period is almost regular and has more than 40 different offsets in the range 0 ms to 16.7 ms.

Estimating latency

We locate the two dots of the paper pattern in the captured image, apply the inverse homography H^{-1} to convert them into display coordinates, apply the *camera-touch* transformation $T(d,\theta)$ to estimate the touch position i of the physical finger at the date of the image d_i.

The last step of our approach is to estimate the date when the event trajectory reaches the physical finger position i. This step is illustrated on Figure 4.

Due to inaccuracies in the calibrations and to non rigid motions of the finger on the paper pattern, the overlap of the event trajectory and the processed image trajectory is not perfect. As a consequence, the event trajectory does not pass exactly through the image position (this is shown on Figure 4 with the event trajectory passing below the image position i). However, as the two trajectories remain very close, we can project the image position on the closest segment of the trajectory. We notice, however, that the trajectory received from the touch display is uneven, resulting in some inaccuracies in the date estimation. However, we are working offline on the whole trajectory data. It is thus possible to reconstruct the smooth trajectory of the finger by fitting analytical functions to the raw events.

For each captured image, we find the segment in the event trajectory that is closest to the finger position i. We then select a sample set of 11 events in the neighborhood of this segment, and we use the RANSAC [4] algorithm to fit a two dimensional 2^{nd} order polynomial $P(t)$ to this sample set.

$$P(t) = \begin{vmatrix} x(t) \\ y(t) \end{vmatrix} = \begin{vmatrix} a_x t^2 + b_x t + c_x \\ a_y t^2 + b_y t + c_y \end{vmatrix} \quad (1)$$

We compute the tangent of $P(t)$ as $dP(t)/dt$, rotate it by $\pi/2$ to get the normal, and then solve for the date $t = d_e$ which correspond to the projection of the finger image position i on $P(t)$ along the normal. d_e is the estimation of the date at which the event trajectory reaches the physical position of

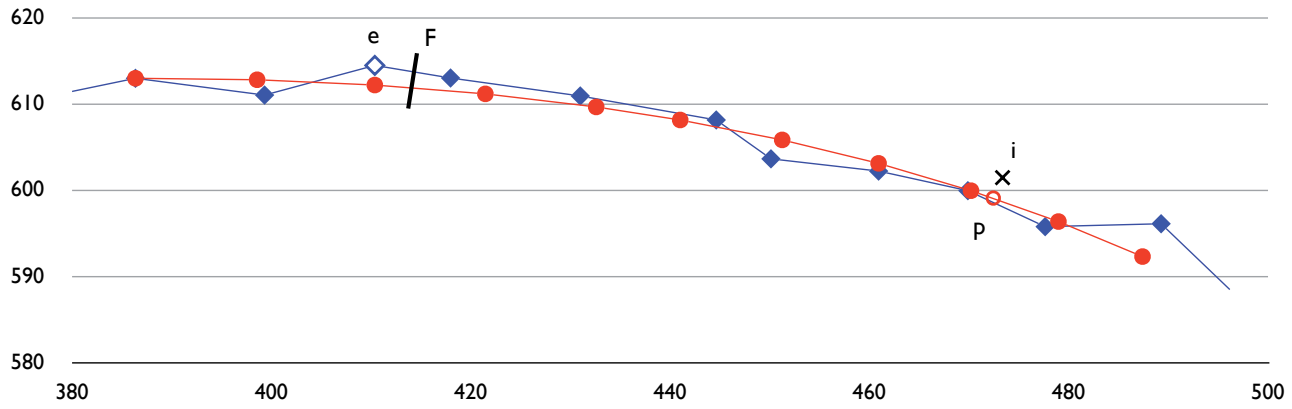

Figure 4. Estimating latency. Abscissa and ordinate represent the x and y position (pixels) of a finger moving from left to right. A two dimensional 2nd order polynomial curve (red discs) is fit to 11 points of the event trajectory (blue diamonds). The position of the finger extracted from the image (black cross i) is projected on this curve (red circle p). Latency is the difference between the date of p and the date of the frame F (illustrated as a small black segment) that was shown on the touch surface when the image was acquired. (e) is the most recent event defining the touch position in the frame F.

the finger. The fit of an analytical function to the raw events yields the estimation of this date at a greater precision than the event sampling period. Finally, we get our latency estimation as:

$$latency = d_e - d_i \tag{2}$$

In the "Experiments" section below, we discuss a set of experiments that we ran in order to assess the accuracy of our approach.

LOW OVERHEAD LATENCY ESTIMATION

The HA approach is our best effort to get the most accurate estimation possible, but it requires a camera, a paper pattern attached to the finger, and several calibration steps.

In this section, we introduce a Low Overhead (LO) latency estimation technique. This technique does not necessitate any additional hardware and allows latency estimations in a matter of seconds.

Idea

Whatever the technique used to measure the latency, two pieces of information are needed: the actual spatiotemporal trajectory of the finger, and the corresponding trace as acquired by the input device and displayed by the output device. The latter part is easy to get: any interactive application can subscribe to the system cursor events, and any touch device can be setup to control the system cursor in absolute mode. The first part is the one that has proved difficult, and required the use of additional hardware (e.g. a camera to track the finger) and some means to associate the event trace with the actual position of the finger.

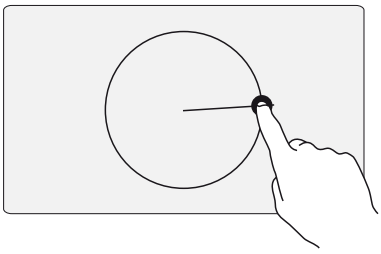

Figure 5. Low Overhead (LO) estimation. Directions given by the system to prescribe the finger trajectory. The radius of the wheel is spinning at constant speed, and the user should keep a finger on the black disk located at the intersection of the circle and the radius.

To eliminate the difficulties of the first part, we turn the problem around: we let the interactive application specify the finger trajectory, and rely on the user to follow its indications. If users are able to precisely follow the trajectory prescribed by the application, then both the actual finger trajectory and the event trace are fully known by the application and can be directly compared to estimate the end-to-end latency.

Implementation

After various experiments, we chose a simple trajectory: moving at constant speed along a circle. The system displays a circle of radius 420 pixels (11.4 cm) centered in the screen. It also displays one radius spinning at constant speed $s = 3\,\mathrm{rad\,s^{-1}}$ (about 2 seconds per revolution). A disk materializes the intersection of the circle and its radius. We call that display *the wheel* (Figure 5). The user is then instructed to follow as precisely as possible the movement of this intersection. Since this movement is highly regular, we expect that people are able to synchronize their hand movements with the wheel and superimpose precisely their index with the disk. In addition, we expect that people are able to make self-assessments about their respect of the prescription.

We compute a latency at each display refresh. We use the most recent touch event. The event provides the position of the finger. Once converted from the Cartesian screen coordinates into the wheel polar coordinates, this position yields the angle α_e of the ray coming from the center of the wheel and passing through the event position. The event was received while the display was showing the prescription at an angle α_p, memorized at the previous display refresh. We compute the latency as:

$$l = (\alpha_p - \alpha_e)/s \tag{3}$$

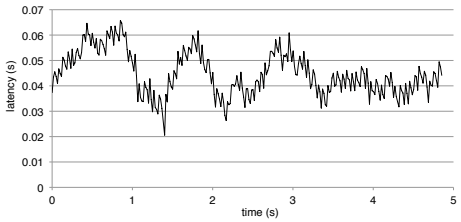

Figure 6. Example of latencies measured with the LO approach.

This measure of the latency is made for every frame, i.e. at 60 Hz. It means that for each revolution of the wheel lasting about 2 seconds we get 125 samples of the latency. By plotting the latencies of the recordings, such as illustrated on Figure 6, we observe that the measured latencies have areas of higher latencies and areas of lower latencies. These areas last for several display frames, hence they can not be explained by the variation of the system's end-to-end latency. We assume that these variations are due to the user being sometimes late and sometimes early with respect to the prescription. This indicates that users are not able to trace a perfect circle at strictly constant speed, which is not a surprise. We expect, however, that user's variations are neutralized on average. This will be shown by the experiments presented in the next section. The requirement to average the measurements, however, implies that the LO approach is not meant to provide a detailed picture of the latency distribution, but rather a simpler latency average.

EXPERIMENTS

In order to assess the validity of our approaches, we ran a set of experiments to compare the latency of a HA system, a LO system, and the "high speed recording" (HSR) technique used by Ng et al. [10]. All approaches were implemented using the same hardware setup, and the graphical rendering was implemented within the same software in order to insure that we were comparing the same end-to-end latency.

Hardware Setup

Our system runs on 3.4 GHz Intel Core i7 processor, and uses an NVIDIA GeForce GTX 680MX graphic card. As a touch-sensitive display, we use a Wacom Cintiq 24HDT multi-touch and pen display. The display has 1920×1200 pixels and a sensor grid of definition 129×83 sampled at 103 Hz. After a first set of experiments, we noticed that the Wacom driver was performing some advanced processing on the signal in order to smooth the reported touch positions. This results in varying latencies depending on the *speed* of a moving finger. The driver also provides an API that gives access to the unprocessed 2D signal of the device. We chose to work with this raw signals in order to get a *finger speed independent* and *minimal* latency, but at the cost of reduced spatial stability. Touch positions are computed as the center of gravity of peaks in the 2D signal, which results in a somehow jaggy trajectory, as illustrated on Figure 4.

Image capture is done with an AVT Marlin F131B camera connected to the computer with a IEEE1394 400 Mbit/s link. The camera is equipped with a CMOS sensor, which allows readout of only subparts of the sensor to achieve higher image rates. In the HSR experiments, the camera is set to capture images of 440×180 pixels at 242 Hz. The camera is set on a tripod and captures an area of 108×44 mm of the touch display as viewed from above. In the HA approach, the camera is set to capture images of 900×600 pixels at 49.7 Hz. The camera is set in front of the touch display, at about 40 cm. It captures a 22×15 cm region of the display, including about 825×550 display pixels. The camera is not used in the LO approach.

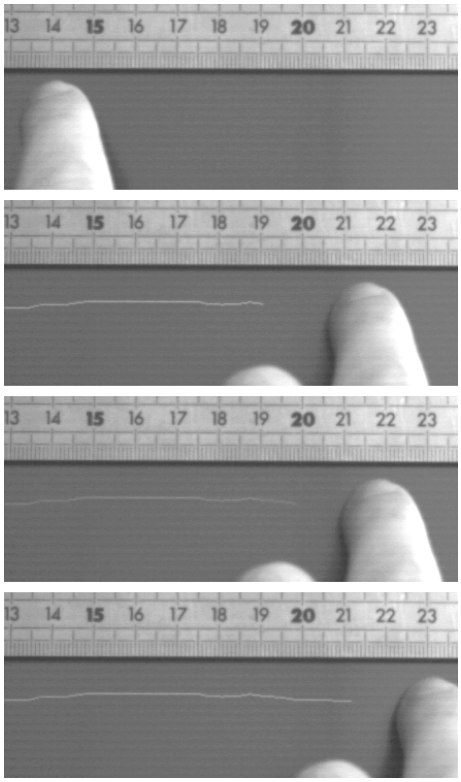

Figure 7. High Speed Recording (HSR) approach, after [10]. Images #262, #309, #311 and #319 of a 242 Hz video. The top and bottom images are used for speed estimation. The two middle images show maximal and minimal gaps between the graphical feedback and the finger.

High Speed Recording (HSR) experiments

We replicate the approach from Ng et al. [10]: we put a high precision ruler on the touch surface in the field on view of the camera, as illustrated in Figure 7. We record a finger sliding along the ruler on the surface at approximately 40 cm/s. We analyze the recorded video image by image in order to isolate the two images where the finger is entirely visible a) just after entering the field of view, and b) just before leaving the field of view. We use the ruler to estimate the finger displacement as 90 mm in 57 camera images at 242 Hz, hence we estimate the average speed at 383 mm/s. We then locate two images where the gap between the finger and the display feedback is maximal and minimal. Using the speed estimation, we convert the gaps to a maximal latency of 61 ms and a minimal latency of 47 ms, which averages to 54 ms.

Our finger position estimations are based on the center of the finger nail, which has been shown to be what users aim

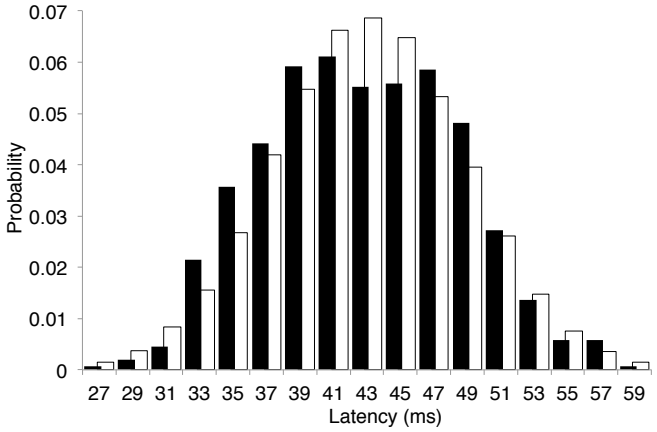

Figure 8. Histogram of estimated latencies with the High Accuracy approach (plain bars). Histogram of normal distribution with identical mean and standard deviation (hollow bars), for comparison.

with [5]. We ignore, however, the actual position where the touch is registered by the display, hence our estimations probably includes an offset to the actual sensing position. This is not a problem for the speed estimation because the difference of finger positions neutralizes the offset. The offset, however, will be the source of sensibly different estimates for the gap length, and thus the source of different latency estimates. Using the left side of the finger on the same images, we obtain a maximal latency of 47 ms, a minimal latency of 34 ms, an average latency of 40.5 ms. When using the right side of the finger, the maximal latency is 76 ms, the minimal 63 ms, and the average 69.5 ms.

High Accuracy (HA) experiments

We make several recordings of the finger attached to the paper pattern, sliding on the surface. A typical recording lasts about 18 s and provides around 770 processed images and latency estimations. All the recordings provide estimates of the average latency in the range 42.5 ms to 43 ms. A typical range of latencies in a single recording is 28 ms to 59.5 ms with a standard deviation of 5.76 ms.

The range of observed latencies is about 31 ms wide. With a known input sampling of 103 Hz and frame rate of 60 Hz, we can explain a range of width 26.4 ms: the minimal latency is observed just after a display refresh where the input had just been received, the maximal latency is observed at the end of the refresh (16.7 ms) when the input was *about* to be updated (9.7 ms). The additional 4.6 ms of observed range may be explained by other sources of variability such as the OS scheduling.

Figure 8 shows a histogram that illustrates a typical distribution of latencies. A Shapiro-Wilk test on this distribution reveals that it does not follow a normal distribution (W=0.994, $p < 0.01$). Indeed, the distribution presents a plateau around the average. We interpret this as follow. Latency probability should be uniform in an interval of the size of the display refresh period (16.7 ms) centered on the average: the probability of making an observation of latency at any point of the display refresh is constant. The further latencies are observed from this interval, the less probable they are as they require the conjunction of less and less probable events such as observing latency just after a display refresh on an event that was just updated.

We make a series of recordings where we programmatically add controlled delays in the delivery of input events. We test delays of 50 ms, 100 ms and 200 ms. We estimate latencies that are the sum of the nominal latency (43 ms) and the controlled delay, at ± 2 ms.

Low Overhead (LO) experiments

Novice experiment

9 subjects (average age 32) participate in a first experiment aimed at validating the wheel approach. All participants are member of a research group on Human-computer interaction. They are familiar with the concept of a system end-to-end latency, but novice to the usage of the wheel.

We first explain the wheel and its goal to each participant. The participant can then take some time to practice at following precisely the movement of the wheel. Then we explain how to start and stop a recording of a trajectory by pressing and releasing a finger on the display with the second hand. The participant is then instructed to perform recordings of trajectories lasting at least 2 laps of the wheel. After each recording, we ask the judgment of the participant about the quality of the record. First, we ask if the record should be kept at all. We then ask if this particular record is better than all the participant's previous records. When the participant feels that there is not more improvement (usually after 10 to 20 recordings) we move to the second part of the experiment.

In the second part, some measurements of the latency distribution are shown to the participant as numbers drawn at the bottom of the display after each recording. We show the range of variation of the latency, its average and median values, and the standard error characterizing the regularity of the latency observed during the record. We explain the meaning of those values, and let the participant make as many recording as desired before answering our question: from the participant's experience, what is the latency of the system?

The results of this experiment are summarized in Table 1. The first part of the experiment is dubbed "blind" since the partic-

Table 1. Novice estimation of the device latency (all values in ms).

	blind		non-blind	
participant	subj.	obj.	subj.	obj.
1	29.11	32.33	38.5	42.43
2	50.10	44.27	40.5	40.04
3	47.10	48.96	40	40.06
4	40.03	41.87	40	43.44
5	46.22	43.44	32.5	35.43
6	37.25	33.12	42	41.37
7	39.43	44.04	40	41.13
8	40.50	32.88	37	37.44
9	43.06	43.06	37	38.47
upper CI	46.20	45.11	40.77	41.91
mean	41.42	40.44	38.61	39.89
lower CI	36.65	35.78	36.45	38.04
CI width	9.55	9.33	4.32	3.87

ipants could not rely on the statistics to make a judgement about the quality of their trials. The second part of the experiment is dubbed "non-blind". For each part, we report a subjective (subj.) value, i.e. a value that relies on the judgement of the participant, and an objective value, i.e. a value that do not rely on his judgement. The objective value is the average latency of the recording that exhibits the smallest standard error, i.e. where the participant was the most regular. For the blind part, the subjective value is the average latency of the recording selected by the participant as being the best recording. For the non-blind part, the subjective value is simply the latency reported by the participant. The last lines of Table 1 report the mean of those values across participants and the lower and upper bound of the 95% confidence interval (CI).

The reported means are in close agreement, since they differ at most by 2.81 ms. For each part of the experiment, the widths of the CIs are smaller for the objective estimations than for the subjective ones, which advocates for a procedure where the subjective evaluation is not used. The CIs are notably thinner in the non-blind part of the experiment (about 4 ms) than in the blind part (about 9.5 ms). This may be due to a training effect, but it may be a benefit of showing the statistics to the participants: having an objective feedback on their performance may encourage the participants to keep trying at improving the quality of their recordings.

Expert Experiment

In a second experiment, we want to explore the accuracy of the LO approach when operated by a user that has had significant training with the wheel. We thus chose one of the authors as the operator. We make a series of 10 successive recordings without allowing rejection. The 10 latency estimates are in the range 39.8 ms to 43.1 ms with an average of 41.2 ms and a CI of 40.4 ms to 42 ms.

We also make a series of recordings with programmatically added delays as with the HA experiments. In this case, the touch events are setup to include an additional random delay, unknown to the operator, in the range 50 ms to 300 ms. For each delay, the operator makes a *single* estimation but includes around 6 revolutions of the wheel. This procedure is repeated 10 times, the whole experiments lasts about 15 minutes. Table 2 shows the setup latency (random delay plus the 43 ms nominal latency of our device estimated with the HA approach), the latency estimated by the operator, and the magnitude of the difference between the two.

Table 2. Expert estimation of controlled latencies (all values in ms).

setup	estimated	abs(difference)
93	96	3
289	291	2
182	183	1
323	321	2
123	126	3
256	258	2
256	257	1
316	321	5
162	170	8
286	290	4

The average magnitude of difference is 3.1 ms, with a standard error of 0.64 ms. The higher bound of the error CI is at 4.42 ms.

DISCUSSION

The HA and LO approaches are in close agreement, providing estimates of the average latency of our system at, respectively, 43 ms and 39.9 ms, less than 4 ms apart. All but one participants of the LO experiment, as well as the expert, provided an estimate a few milliseconds lower than the HA estimate. This may indicate that our implementation of the LO approach introduces a small systematic negative bias.

The HSR experiment illustrates the imprecision of the approach as, depending on where the touch location is chosen relative to the finger, it provides estimates in the range 40.5 ms to 69.5 ms, a rather large 29 ms interval. The HA estimate falls within this interval, showing that both approaches are in agreement. The HA estimate is on the lower side of the HSR range, suggesting that the touch location is on the left side of the finger. This is coherent with the images of Figure 7 where the finger appears to be leaning to the left.

The HA approach allows to effortlessly collect of a large sample set of latencies, for example 770 estimates in a recording lasting 18 s. This is a major improvement over previous approaches that either relied on the manual labeling of the recorded data, or provided automated estimation but for a single estimate per recording. The large sample set provided by the HA approach increases the probability of observing the true extremums of latencies exhibited by the system. In addition, it provides a detailed picture of the distribution of latencies, revealing a non Gaussian distribution with a plateau corresponding to the display period. This kind of precision opens the way to novel techniques aimed at reducing latency, for example by improving in software the synchronization between the sensing and the display.

The HA approach has already proven useful to estimate the accuracy of the LO approach. We were surprised to see how close to the ideal novice participants were able to get: all participants but one estimated latency less than 5 ms away from the HA estimate. The experiment with an expert operator showed a strong stability of the approach with a 95% confidence interval of width 1.6 ms. It also showed a surprisingly good accuracy with an average estimate only 1.8 ms away from the HA estimate.

The LO approach offers both precision and accuracy in the order of 4 ms, which is on par with, if not better than, previous approaches. This is achieved without any equipment (e.g., a camera) in addition to the touch surface, and the measurement is performed in a matter of seconds. We hope that this approach will contribute to the widespread adoption of the end-to-end latency evaluation in interactive touch systems. This should prove particularly useful for designers to evaluate the influence of their designs on the latency of the system.

We recommend to use the non-blind protocol and to retain the recording with the lowest standard error, since it produced latencies having the smallest variability. Also, the LO expert experiment with random controlled latencies seem to show

an increasing difference between controlled and estimated latencies on the last 3 runs of a short 15 minutes session (see Table 2). This seems to indicate a fatigue effect. Indeed, sliding a finger in perfect circles at constant rotational speed is not an easy task as it involves the proximal joints of the elbow and shoulder. When estimating the latency of a single particular device, this should not be a problem. If many estimations are required, the LO approach will be more accurate by allowing some resting time between estimations.

CONCLUSION AND FUTURE WORK

In this paper, we introduce two novel approaches to the estimation of touch systems end-to-end latency. Thanks to its automated processing of the captured images, the High Accuracy (HA) approach provides a high number of accurate latency estimations with little effort from the experimenter. Using this approach, we provide the first detailed image of the distribution of latencies in a typical interactive touch system. We also introduce a Low Overhead (LO) approach. It also includes automated data processing, and it is the first approach that does not require any external equipment for latency estimation, thanks to the cooperation of a human operator. In a series of experiments, we show that novice operators of the LO approach are able to provide estimations no further than 4 ms from the HA estimate. We hope that the LO approach will open the way to the widespread characterization of touch system's latency.

We see several extensions to this work. Adapting the HA and LO approaches to indirect input devices, such as a computer mouse or a 3D tracker, should be possible by using a projected graphical display. Indirect devices can be made "direct" by sliding them on the projected display, provided that their reported position are correctly calibrated with the display.

The LO approach could be improved in different ways. One of them is facilitating the operator's task. The circular trajectory requiring the complex coordination of the elbow and the shoulder, it could be adapted to require a simple forearm oscillation for example. Also, the effect of the training of operators was not studied in this work, it may suggest ways to improve the accuracy of the approach.

ACKNOWLEDGMENTS

This work was partially funded by the French government in the FUI project 3DCI (AAP14).

REFERENCES

1. Adelstein, B. D., Johnston, E. R., and Ellis, S. R. A testbed for characterizing dynamic response of virtual environment spatial sensors. In *ACM Symposium on User Interface Software and Technology (UIST)*, ACM (1992), 15–22.
2. Bi, X., Li, Y., and Zhai, S. FFitts law: Modeling finger touch with Fitts' law. In *ACM Conference on Human Factors in Computing Systems (CHI)*, ACM (2013).
3. Brainard, D. H. The psychophysics toolbox. *Spatial Vision 10*, 4 (1997), 433–436.
4. Fischler, M. A., and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6 (1981), 381–395.
5. Holz, C., and Baudisch, P. Understanding touch. In *ACM Conference on Human Factors in Computing Systems (CHI)*, CHI '11, ACM (New York, NY, USA, 2011), 2501–2510.
6. Jota, R., Ng, A. N., Dietz, P., and Wigdor, D. How fast is fast enough? a study of the effects of latency in direct-touch pointing tasks. In *ACM Conference on Human Factors in Computing Systems (CHI)*, ACM (2013).
7. Liang, J., Shaw, C., and Green, M. On temporal-spatial realism in the virtual reality environment. In *ACM Symposium on User Interface Software and Technology (UIST)*, ACM (1991), 19–25.
8. MacKenzie, I. S., and Ware, C. Lag as a determinant of human performance in interactive systems. In *ACM Conference on Human Factors in Computing Systems (CHI)* (1993), 488–493.
9. Mine, M. R. Characterization of end-to-end delays in head-mounted display systems. Tech. rep., Chapel Hill, NC, USA, 1993.
10. Ng, A., Lepinski, J., Wigdor, D., Sanders, S., and Dietz, P. Designing for low-latency direct-touch input. In *ACM Symposium on User Interface Software and Technology (UIST)*, ACM (2012), 453–464.
11. Pavlovych, A., and Stuerzlinger, W. The tradeoff between spatial jitter and latency in pointing tasks. In *ACM SIGCHI symposium on Engineering Interactive Computing Systems (EICS)*, ACM (2009), 187–196.
12. Steed, A. A simple method for estimating the latency of interactive, real-time graphics simulations. In *ACM Symposium on Virtual Reality Software and Technology (VRST)*, ACM (2008), 123–129.
13. Swindells, C., Dill, J. C., and Booth, K. S. System lag tests for augmented and virtual environments. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, ACM (2000), 161–170.
14. Teather, R. J., Pavlovych, A., Stuerzlinger, W., and MacKenzie, I. S. Effects of tracking technology, latency, and spatial jitter on object movement. In *IEEE Symposium on 3D User Interfaces (3DUI)*, IEEE Computer Society (2009), 43–50.
15. Ware, C., and Balakrishnan, R. Reaching for objects in VR displays: lag and frame rate. *ACM Transactions on Computer-Human Interaction (TOCHI) 1*, 4 (1994), 331–356.
16. Watson, B., Walker, N., Ribarsky, W., and Spaulding, V. Effects of variation in system responsiveness on user performance in virtual environments. *Human Factors: The Journal of the Human Factors and Ergonomics Society 40*, 3 (1998), 403–414.