Automating Latency Measures for Touch Screen Input

by

Sunyoung Park

Thesis

Submitted to the

Department of Computer Science of Charleston Southern University

for the degree of

Master of Science

in

Computer Science

May, 2023

Table Of Contents

CHAPTER I

Introduction

Latency in a system’s response to user input adversely impacts user performance. Therefore, system designers need an approachable and effective way to measure such latency to determine its impacts. However, current approaches require additional equipment or much manual work. These requirements limit the usability of the methods. A study shows that any increase in latency decreases user performance, and users may be able to perceive latency as low as 1 ms (Jota et al., 2013). An effort to accurately and easily measure the latency will be considerably helpful for enhancing the latency performance to improve user experience. Therefore, I introduce an automated latency measuring tool (LMT) using an object-tracking algorithm. The LMT measures latency using both the distance-measuring and frame-counting techniques. The LMT automatically measures the distance between the stylus position and the tracking graphic by converting the number of pixels to physical distance. The velocity of the stylus movement is then used to estimate the latency. To support the frame-counting technique, the LMT logs the locations of the stylus and graphic trajectory and calculates latency using frame number difference. After conducting a human performance experiment and performing manual analysis to validate the accuracy of the LMT, it was found that the LMT is reliable and capable of producing precise latency measurements for both drawing and tapping tasks.

I.1 Measuring Latency

The latency has a significant impact on user performance. A study found that user performance increased as the latency became lower, and users could perceive touchscreen latency as low as 2.38ms (Ng et al., 2012). Because of the significant impact of latency, many techniques have been adopted to measure the latency accurately, including the Time measuring, Frame counting, Distance measuring, and Sine-fitting techniques (Bérard & Blanch, 2013; Bockes et al., 2018; Deber et al., 2016; Kaaresoja & Brewster, 2010; Ng et al., 2012; Pavlovych & Stuerzlinger, 2009; Steed, 2008; Teather et al., 2009; Vrouwenvelder et al., 2021; Yun et al., 2017). However, system designers often overlook latency when evaluating a system because it is difficult or expensive to measure with any level of accuracy. Many systems have been proposed to measure the latency, which are discussed in section 2.4. However, most systems require extra hardware, manual work, or custom software. These limitations make the latency measuring process expensive or time-consuming. To overcome these limitations, I developed the LMT, which offers both convenience and affordability.

The details of the commonly used techniques are described in Section 2.2. Section 2.4 presents the details of the currently proposed systems and their limitations.

I.2 Approach

Considering the limitations of current systems, I created an automated latency measuring tool, LMT, using an object tracking algorithm. The LMT measures the distance between the stylus position and the tracking graphic by converting the number of pixels to physical distance. With an estimation of the velocity of the stylus, an evaluator can achieve the measured latency. The LMT also gives the number of frames the tracking graphic takes to reach the location of the stylus. By giving both the physical distance and the number of frames, evaluators can choose either the distance-measuring method or the frame-counting method to measure the latency. The LMT requires no hardware or software except an external camera that records the target touchscreen. The LMT eases the burden on evaluators by making the measuring process automated. After an evaluator inputs the actual screen size and clicks the four edges of the target screen, the LMT outputs a calculated latency. The LMT also supports both tapping and dragging interactions. However, evaluators may need to place a mirror beside a screen to capture the exact moment a finger or a stylus touches the screen.

I.3 Research Summary

The design of the is presented in Chapter 3, followed by a detailed description of the accuracy validation process in Chapter 4. Chapter 5 presents the results of the accuracy analysis and Chapter 6 discusses the key findings, and the limitations of the LMT, and provides recommendations for future improvements.

CHAPTER II

Literature Review

II.1 Impact of Latency

The direct-touch interface latency is a significant factor in user performance degradation. According to Ng et al. (2012), commercial touch screens typically take 50-200ms to update the display in response to a physical touch action. This latency represents the physical distance between the touch position on the touchscreen and the tracking graphic on display.

Ng et al. (2012) constructed a direct-touch device that achieves 1 ms of latency to test human perception of drag latency on a touch screen. By testing with a range of 1 ms to 65 ms of latency, the researchers found that user performance increased as the latency became lower, and users could perceive touchscreen latency as low as 2.38 ms.

Jota et al. (2013) followed a similar approach to Ng et al. (2012), but focused on finding a performance floor. The level of latency varied from 1 ms to 50 ms, and various target widths and the distance between the starting point and the target were applied. The researchers found a significant effect for the interaction between distance, width, and latency. Also, they found evidence that any increase in latency decreases performance, and users may be able to perceive latency as low as 1 ms.

II.2 Current Techniques

With the increasing use of touchscreen technology, much research has been conducted to measure the latency of direct-touch interfaces or the effect of latency on user performance.

Estimating the latency requires many considerations because the latency has many sources. For instance, the latency varies depending on the device's touch sampling and refresh rate. The touch sampling rate is the frequency at which the device's display reads the input, and the refresh rate controls the frequency of pixel change. These rates vary from device to device. Furthermore, additional latency takes place from applications or network status.

One of the widely used techniques is the frame-counting method. This technique usually requires an external camera to capture the misalignment between a touch point and a tracking graphic on display. The latency can be measured with the difference in the number of frames between the current frame and the frame that the tracking graphic reaches the current touch location of the touch point (Kaaresoja & Brewster, 2010; Pavlovych & Stuerzlinger, 2009). Another widely used technique is directly measuring the time difference without counting the number of frames. Some researchers used this technique by measuring the time between the input event and the output event or the time that the device's sensor catches the input signal (Bérard & Blanch, 2013; Bockes et al., 2018; Deber et al., 2016; Jota et al., 2013; Teather et al., 2009; Vrouwenvelder et al., 2021; Yun et al., 2017). The latency can also be estimated by measuring the physical distance between the position of the touch point and the current tracking trajectory. However, this technique requires the estimated velocity of finger movement (Bérard & Blanch, 2013; Ng et al., 2012; Yun et al., 2017). While these three techniques are commonly used, Steed (2008) introduced a new approach called the sine-fitting method, which uses two sine curves from a tracker and a simulated image to estimate the latency.

II.3 Limitations

There are many systems, which are discussed in Section 2.4, that adopt these techniques to measure the latency of the touchscreen. However, each system shows several limitations. First, they require extra hardware or specially designed tools, limiting usability. Second, the number of samples is limited because the systems necessitate manual work. Third, some systems do not support generalizability in the use of software because they use customized software. With software requirements, users cannot use preferred applications or devices. Lastly, some systems do not support flexibility in interaction types because they specialize in a single interaction type, such as tapping or dragging.

II.4 Latency Measuring Systems

Most previously proposed systems for latency measurement are based on one of the techniques mentioned in Section 2.2. However, each experimental setup or requirement differs from the others. Table 1 describes the requirements for each system that are addressed in this chapter. Systems included in end-to-end latency measure latency from physical input to screen output. Those are not included in the category measure latency from physical input to when the system recognizes the touch. Systems that require extra hardware, except a camera, are included under the custom or non-custom hardware requirements category. Software requirement means the latency measurement is constrained with a specific application. As shown in Table 1, half of the 12 systems require manual work, and eight of them require extra hardware setup except an external camera. Every system in Table 1 requires either manual work or extra hardware/software. More details, advantages, and limitations of each system will be discussed in the following chapters.

Ref.	Input Device Types	Latency Type	Interaction Types		Method	Manual Work	Hardware Requirements			Custom Software Requirements
		End-to-End	Tapping/ Clicking	Dragging			External Hardware	Device Dependency	External Camera	
Bérard & Blanch, 2013 **(HA)**	Direct			✓	Time Measuring		✓		✓	✓
Bérard & Blanch, 2013 **(LO)**		✓		✓	Distance Measuring					✓
Bockes et al. (2018)	Indirect		✓		Time Measuring		✓			
Deber et al. (2016)	Direct	✓	✓		Time Measuring		✓			
Kaaresoja & Brewster (2010)	Direct	✓	✓		Frame Counting	✓			✓	
Ng et al. (2012)	Direct	✓		✓	Distance Measuring	✓	✓		✓	
Pavlovych & Stuerzlinger (2009)	Indirect	✓		✓	Frame Counting	✓			✓	
Schmid & Wimmer (2021)	Indirect	✓	✓		Time Measuring		✓			
Schmid & Wimmer (2023)	Indirect		✓		Time Measuring		✓			
Steed (2008)	Direct	✓		✓	Sine Fitting Method		✓		✓	✓
Teather et al. (2009)	Indirect	✓		✓	Time Measuring	✓	✓		✓	
Vrouwenvelder et al. (2021)	Direct		✓	✓	Time Measuring		✓			
Yun et al. (2017) **(Indirect)**	Direct	✓		✓	Time Measuring	✓	✓			
Yun et al. (2017) **(Direct)**		✓		✓	Distance Measuring	✓			✓	

Table 1. Summary of the latency measuring systems that are addressed in Section 2.2

II.4.1 Using Time-Measuring Technique

Bérard & Blanch (2013) designed a novel “High accuracy” (HA) approach and “Low Overhead”(LO) approach to estimate the latency of touch screens. The HA approach uses the time-measuring method, and the LO approach uses the distance-measuring method to estimate latency. HA approach, which requires an external camera and careful calibration, utilizes the recorded trajectory as the time measurement component. HA automatically counts image frames and shows the current frame number on display. The current frame number is then utilized to find the event time in the frame log. The time the trajectory reaches the observed physical touch position is then found in the event log to calculate the time difference. This approach estimates latency between the physical touch event and input log event, which differs from end-to-end latency. The authors used a visual pattern attached to the finger to allow the extraction of an accurate touch position from images of the finger. This method could effectively decrease the possibility of having an inaccurate result. Because the method is automated, users can measure the latency with one video. However, this method requires careful calibration.

Another research used the time-measuring to estimate the latency of USB-connected devices with button presses(Bockes et al., 2018). The study limited the experiment to the latency of binary input channels. The authors presented a tool based on a Raspberry Pi 2 microcomputer that toggles a button and automatically captures the time the input signal arrives. Because measuring the exact time of clicking the button is hard, they solder two wires to the input device to simulate a button press without physically clicking it. These two wires are connected with an audio jack to the phototransistor side of an optocoupler. The optocoupler then connects to one GPIO pin, which generates electricity to simulate clicking the button. An application running on the RPi2 controlled generating triggers and calculating the latency. The authors conducted validation tests with eight devices and collected 5000 samples for each. The results addressed that devices have different latency ranges and various latency distributions. By soldering two wires to the input device, the authors could get more sample data than the amount of data they could get by physically clicking the button. This method benefited the authors because many samples can enhance the quality of an experiment result. However, their method only supports indirect devices and the latency from the binary input channel.

Schmid & Wimmer (2021) proposed a method for measuring the end-to-end latency of traditional setups with a button-equipped input device and a display. By automatically triggering the buttons of modified input devices, they controlled the moment when an input event was triggered and detected a screen output using a photo sensor attached to the screen. This fully automated measurement process is accomplished through a microcontroller, making it highly efficient. While this method is limited to indirect input devices like a mouse or keyboard and requires external hardware, evaluators can take advantage of this approach to expedite the assessment process. This advantageous feature has prompted another study (Schmid & Wimmer, 2023) to adopt a modified version of this method as part of their study to measure the latency of graphic frameworks, which differs from the end-to-end latency.

Deber et al. (2016) proposed an automated tool for measuring touchscreen device interface latency called Latency Hammer. The hammer consists of the measurement head, which contains a photodiode and brass contact, and the main board, which controls a microcontroller. The measurement head is located on the top of the touchscreen device, and all the steps can be automated without human interaction. The measurement head generates the simulated touch and delivers it to the screen through a brass contact. A photodiode, located on the bottom of the head with a brass contact, captures display brightness from the screen. The main board is responsible for triggering the head for generating a touch and sensing the photodiode for changes in brightness. After generating a touch and getting brightness feedback, the software in the Hammer calculates the time difference between those two events and outputs the latency. To validate the accuracy of the Hammer, the authors conducted five validation tests with different approaches. In one of the tests, a customized application was utilized to generate various delays, and the authors measured the added latency with the Hammer. As a result, these validation techniques all proved the reliability of the Hammer. Because the Hammer requires no human involvement, the method can reduce the possibility of human-induced errors during the measurement. However, the method limits usability because users have to have customized hardware to measure latency. The method is also limited to measuring the latency with tapping events only.

Vrouwenvelder et al. (2021) proposed a method to measure dragging latency, which differs from end-to-end latency. This study presents a methodology utilizing a laser and laser sensor to obtain an external measurement of stylus position that is compared with the digitally measured touch event data from the operating system's kernel. The authors placed a laser and a laser sensor on opposite sides of the touchscreen so that the laser sensor could capture the light-to-dark and the dark-to-light events from a stylus while dragging performance is taking place on the touchscreen. However, the placements of kernel touch events were not aligned with the laser beam, representing the physical position of the stylus on the screen. This phenomenon is due to drag latency and the input pattern. Therefore, the latency was measured as the time shift required to align the locations of touch sensor activation with the recorded laser beam crossings. After the validation process, the authors confirmed that touchscreen-drag latency significantly depends on the input speed. The authors used a WALT latency timer with a built-in accelerometer to capture the exact moment of a tap event. By comparing the result from the WALT timer and the registered time of the kernel, the authors could get more accurate data. However, their method requires extra hardware. The method is also limited to measuring the latency between the physical input event and the input log.

Teather et al. (2009) studied the effects of input device latency and spatial jitter on 2D pointing tasks and 3D object movement tasks. The authors used a 3D tracker and a mouse to measure the latency in 2D pointing tasks. They placed a physical pendulum in front of the display. The mouse was affixed to a tripod positioned in front of the pendulum to sense the pendulum's movement. They also used software to draw two lines on the screen. As the pendulum swung, the ends of the two lines moved with the pendulum's motion perceived by the mouse and the tracking system. After recording both the pendulum's movement and the lines, the authors manually examined the video to calculate the time the mouse and the tracker lines take to catch up with the pendulum's movement. This method could be costly because of hardware and software requirements. Furthermore, the requirement for manual analysis can constrain the number of experiments.

Yun et al. (2017) introduced an asynchronous design of the input-to-display path called Presto. Presto can reduce the end-to-end latency below 10ms by eliminating the synchrony in the input to display path. When the authors tested the performance of Presto, two methods were used: direct and indirect. The indirect method estimates the latency using OS-based time logging. This method breaks down the end-to-end latency into three parts: from physical touch to the touch device driver, from the touch device driver to the display subsystem, and from the display subsystem to display externalization. The authors measured the first latency using a microcontroller and two light sensors. When the stylus crosses the first laser beam, the microcontroller captures the change in the light sensor output and logs a timestamp. When the touch device driver receives an event, it turns on the built-in LED. Then the microcontroller detects the change in the second light sensor output and logs a timestamp. The second and third latency was measured by two timestamps and y-coordinate of the touch event logged in software. This method could be effective because the method applies to all applications. However, the method requires hardware setup and manual work.

II.4.2 Using Frame-Counting Technique

Kaaresoja & Brewster (2010) introduced a multimodal latency measurement tool for touchscreen interaction. The authors focused on visual, audio, and tactile feedback latencies. A high-speed camera and mirror were used to detect the visual latency and capture the exact moment a finger or stylus touched the screen. Then the number of frames from the touch event to the visual feedback is calculated for visual latency. They also used an accelerometer to pick up audio and tactile feedback latencies. Transcoding the analog output of the accelerometer into audio through the soundcard resulted in better performance in catching the feedback. Testing four commercial touchscreen mobile phones with their tools confirmed that there are observable latencies in the devices measured. Also, they partly confirmed that the latencies vary between devices, applications, and modalities. The authors placed a mirror beside the screen to have a clear view of when the stylus hits the screen. This simple method gave an accurate result. However, their method only allows measuring the latency with tapping actions. Also, the authors needed to obtain the latency by hand and eye using a slow-motion video player. This method is costly and time-consuming.

Pavlovych & Stuerzlinger (2009) used the frame-counting method to measure the latency of indirect input devices. They used a mouse as an exemplary low-latency and low-jitter device.

They artificially added latency and jitter to match the range of latency and jitter present in other commonly used devices. A video camera filmed the motion of the mouse and the cursor on the screen. To find the average input device delay, the authors averaged ten measurements of two minutes of video each. This study also required manual analysis of the recorded videos. Based on the range of latency values, the authors experimented with human performance with various values of latency, jitter, target width, and target amplitude. As a result, all four independent variables significantly affect user performance. To measure the average latency, the authors repeatedly moved the mouse by hand sideways along the top edge of the display bezel and recorded the mouse and the cursor movement. This method could give an inaccurate latency result because of the high possibility of human-induced error. When people move a mouse sideways, people are prone to stop at the end to change direction briefly. If the cursor movement is much slower than the mouse movement because of high latency, people tend to stay at the end longer because they do not know the exact end point. Therefore, this method is constrained to measuring an accurate latency and the method.

II.4.3 Using Distance-Measuring Technique

Bérard & Blanch (2013) designed the LO and HA approaches. The LO approach does not require external equipment and uses an interactive application to estimate the end-to-end latency. The application displays a target point moving at constant speed along a circle. The user is then instructed to follow the target with a touch position as precisely as possible. Then the latency is calculated with the difference between the angles, the angle at which the touch event happened, and the angle at which the display was showing the target on the screen. However, this method does not give a result as precise as HA because the speed of users' fingers is assumed to be the same as the speed of the moving target. Even though the LO approach is not as accurate as the HA approach, the method is efficient in measuring the approximate latency because it does not require any external equipment. However, this approach limits the generalizability of the system because the use of the specific application is required. Users cannot measure the latency on their devices with their preferred applications.

Ng et al. (2012) proposed a simple approach to measure the end-to-end latency using the distance-measuring method. Because the distance-measuring method requires an estimation of finger movement speed, the authors placed a ruler on the touchscreen. The speed estimation can be achieved by examining the distance a finger traveled between frames. Then the authors measured the distance between the finger and the following graphic to get the final latency value. Using a ruler to estimate a finger's speed is simple and efficient. However, this cannot derive an accurate result because the movement of a finger is assumed to be constant. Also, getting a lot of sample data with this method is costly because manual work is required.

When Yun et al. (2017) tested the performance of Presto, two methods were used: direct and indirect. The direct method measures the end-to-end latency by manually analyzing video records. The authors noted the distance between the location of the stylus and the following trajectory. The latency value was achieved by estimating the velocity of the pen movement. When the authors compared the latency derived from the two methods, the difference was within 2.5 ms, which was smaller than their standard deviation. However, this method cannot derive an accurate result because the stylus speed is not accurate, and the speed is assumed to be constant.

II.4.4 Using Sine-Fitting Technique

Steed (2008) used the sine-fitting method to measure end-to-end latency. The author utilized a pendulum to simulate a continuous movement of input. The latency can be measured by simultaneously capturing a video of a target attached to a tracking device moving on a pendulum and the movement of a simulated image that uses the resulting tracking data. Then the author used the sine-fitting method to estimate the latency. The author extracted position information from the video and reconstructed the motion. Then two sine curves were fitted, one to the horizontal displacements of the tracker and one to the horizontal displacement of the simulated image, to calculate the phase difference between the two. The author argued that the sine-fitting method derived a more accurate latency result than the frame-counting method. The authors tested the method to estimate the latency of two different image generators and could get the expected result of an introduced frame delay. The author asserted that about 4 seconds of video and 3-4 swing cycles were sufficient to get a reasonable latency estimate. Even though this benefit can reduce the time for an experiment, the method still burdens users to prepare additional hardware such as a pendulum and a tracker.

II.5 Conclusion

Based on the literature review, most current systems require extra hardware or software. Hardware requirements could be a bottleneck giving users a burden to prepare the required hardware. Software requirements can limit generalizability because users must use a specific application. As you can see in Table 1, half of the addressed systems require manual work. Manual work requirements could be costly and time-consuming while limiting the number of samples in experiments. Considering these limitations, I created the LMT that does not require any hardware or specific software running on the device except an external camera. The LMT offers flexibility by allowing users to choose their preferred latency-measuring method, either the frame-counting method or the distance-measuring method. The detailed design will be addressed in Chapter 3.

CHAPTER III

Latency Measuring Tool (LMT)

In this chapter, I present the design of the LMT for measuring touchscreen latency. The LMT supports two techniques: the distance-measuring technique and the frame-counting technique. After recording the target screen and stylus movement from a top view, the evaluator needs to upload the camera recording to the LMT. Before the LMT can begin measuring latency, the evaluator is prompted to input the physical dimensions of the target area and click on the four edges of the target screen to calibrate the LMT. This step is necessary for the LMT to recognize the target area and to correctly calculate the pixels-per-centimeter using the evaluator's input that was given earlier. The LMT then utilizes object tracking and color detection algorithms to automate the process of calculating the velocity of stylus movement and the distance between the current stylus location and the graphic trajectory, resulting in the output of the latency value using Eq. (1), where C_t is the current position of the stylus and D_t is the current position of the display feedback. The value of *v*, which represents the stylus velocity, is calculated using Eq. (2), where C_{t-10} represents the position of the stylus 10 frames prior to the current position. To support the frame-counting technique, the LMT logs the locations where stylus movement is detected and calculates the latency by detecting when one of the locations in the log is captured again ($frame\#_{Ct}$ - $frame\#_{Dt}$). The LMT utilizes object tracking and color detection algorithms to automate the whole process. The details of the design and implementation of the LMT are discussed in the following sections.

$$Latency = \frac{Ct - Dt}{v} \qquad \text{(Eq. 1)}$$

$$v = C_t - C_{t-10} / 10\ frames \qquad \text{(Eq. 2)}$$

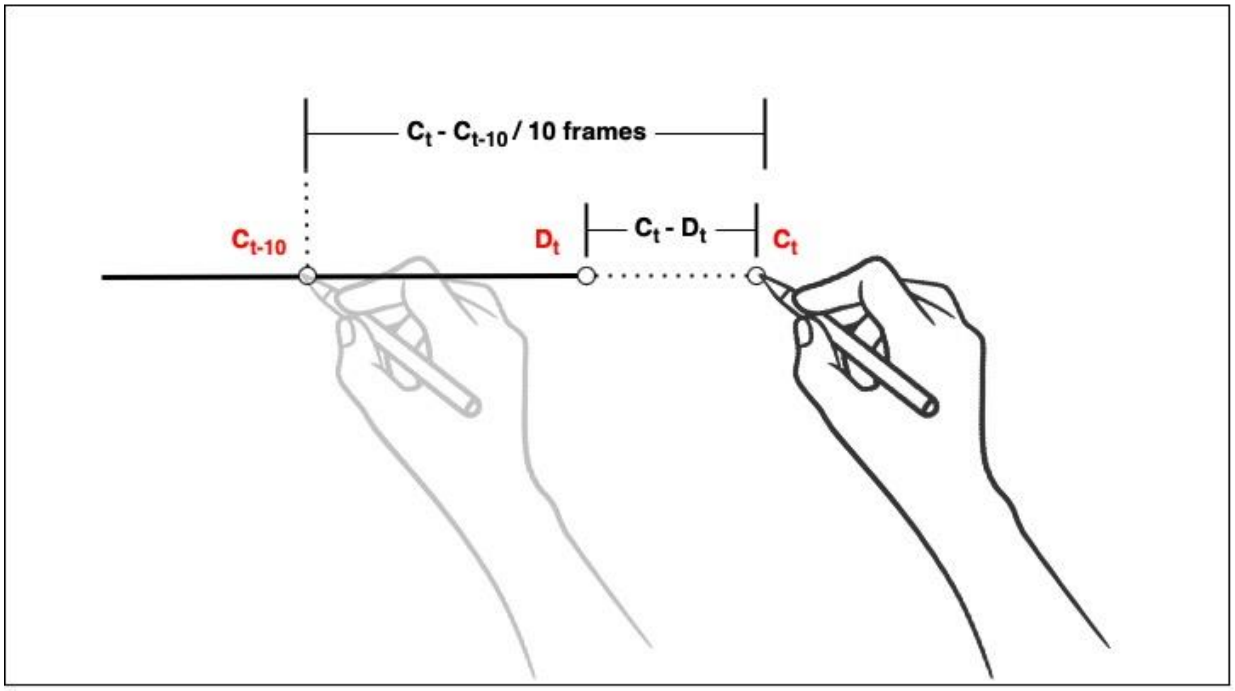


Figure 1: Distance-measuring technique. Latency is calculated by measuring the distance between the current position of the stylus (C_t) and the current position of the display feedback (D_t). The velocity of the stylus is calculated by measuring its travel distance over a period of 10 frames.

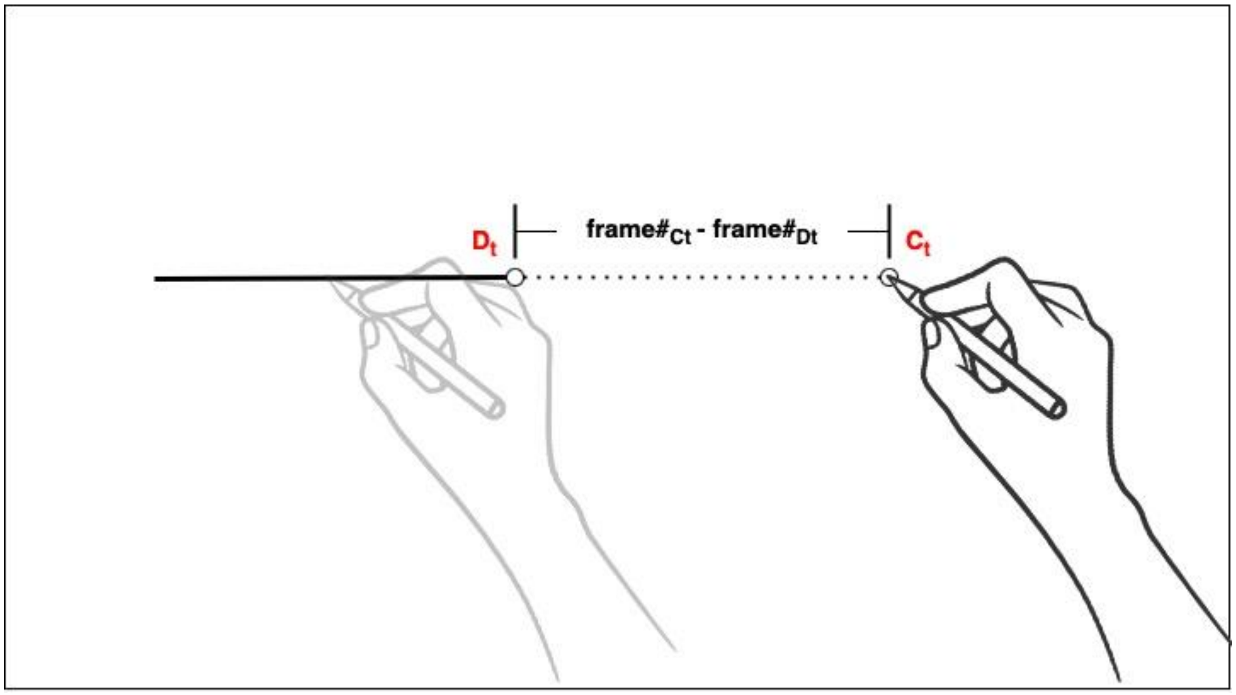


Figure 2: Frame-counting technique. Latency is calculated by measuring the frame number difference between the current position of the stylus (C_t) and the current position of the display feedback (D_t).

III.1 Distance-Measuring Technique

As shown in Table 1, the distance-measuring technique often requires manual measurements to determine the distance between a stylus and a graphic trajectory (C_t - D_t). The LMT offers automatic distance-measuring capabilities utilizing several machine-learning algorithms. Specifically, I leverage an object-tracking algorithm to monitor the movement of the stylus and the graphic trajectory across the screen, while also employing a color detection algorithm to differentiate between the two. This approach enables us to accurately estimate the velocity of the user's dragging motion and the distance between the stylus and the graphic trajectory. To compute the stylus's velocity (v), I record the locations of the stylus in each frame and use the formula presented in Eq. (1) to calculate the latency value for each frame.

III.1.1 Pixels per Centimeters Measure

To accurately measure distances in a video, the LMT requires calibration to convert pixel measurements to physical distances. To perform calibration, the user is prompted to input the physical dimensions of the target screen and click on its four edges. The LMT references the dimension values to calculate the pixels-per-centimeter ratio. For example, if the target screen has an actual dimension of 15.75 cm and is computed as 942 px in the video frame, the pixels-per-centimeter ratio is calculated as 59.8 px, indicating that there are approximately 59.8 px for every 15.75 cm in a video frame. This ratio is used to convert pixel measurements to their corresponding physical distances. The resulting values are used to calculate the physical distance between the stylus and graphic trajectory and the distance the stylus moved over the previous 10 video frames to determine its velocity.

The choice of 10 frames as the basis for velocity measurement in the LMT was motivated by practical considerations. In particular, many common camera frame rates, such as 60 fps and 240 fps, are easily divisible by 10, which simplifies calculations and reduces the likelihood of rounding errors.

III.1.2 Object Tracking Algorithm

The LMT is implemented in Python using the OpenCV library. I utilized a centroid tracking algorithm for object tracking, which operates based on the Euclidean distance between object centroids. In each frame, the algorithm calculates the centroid of detected objects using their bounding box coordinates. It then assigns an ID to each bounding box and calculates the Euclidean distance between every pair of centroids. The algorithm assumes that the same objects in the current and subsequent frames will have the minimum distance. Therefore, the two pairs of centroids with the minimum distance in subsequent frames are considered the same object and assigned the same ID.

The algorithm then measures the distance between each coordinate between two different objects. For instance, Object 1 and Object 2 will have top-left, top-right, bottom-left, and bottom-right coordinates, respectively. Then the LMT connects every pair of the coordinates of the two objects, resulting in 16 pairs. The algorithm assumes that the distance between the stylus and the following graphic trajectory will have the minimum distance among all the pairs. Thus the shortest distance is selected and converted to the actual dimension to be used as a final output.

In the extended version of the LMT, I calculate the distance between the closest coordinate of the following trajectory to the stylus and all four coordinates of the bounding box of the stylus. This approach ensures that users can still obtain the latency value even when the detected shortest path is incorrectly calculated.

III.1.3 Color Detection Algorithm

The objects detected using the object-tracking algorithms are classified into two categories, the stylus and tracking trajectory, based on their colors. I used a green color cap on the tip of the stylus and set the color of the graphic trajectory to blue. By using different ranges of color masks, the LMT can differentiate between the objects. This approach not only helps in estimating the velocity of the stylus movement but also in accurately determining the touch point. Detecting the exact touch point while drawing on the screen using a stylus or finger can be challenging because the touch point can be hidden by the stylus or finger. However, using a specific color on the tip of the stylus or the top of the fingernail mitigates this problem.

III.1.4 Finger/Stylus Speed Estimation Support

The LMT automatically measures the distance the stylus has traveled and calculates the time using the camera frame rate. To accurately measure the latency and eliminate the need for manual work in the distance-measuring technique, the LMT must also automatically calculate the stylus speed. The speed can be achieved using Eq. (2), where C_t denotes the current position of the stylus and C_{t-10} represents the position of the stylus 10 frames prior to the current position. However, because the stylus movement cannot maintain a constant speed, the LMT logs the stylus locations in each frame and calculates the speed using the change in position in the last 10 recorded frames. The recorded framerate is then used to determine the time elapsed.

III.2 Frame-Counting Technique

The frame-counting technique can measure the latency using the difference in the number of frames between the current frame and the frame when the tracking graphic reaches the current touch location of the finger or stylus (Kaaresoja & Brewster, 2010; Pavlovych & Stuerzlinger, 2009). However, as shown in Table 1, this technique often involves manual work, which is both time-consuming and expensive. In contrast, the LMT automates the location note and frame-counting process using the same algorithms used to support the distance-measuring technique. Additionally, I use perpendicular line calculation to expand the user's input area to account for any potential offsets when capturing the moment the graphic trajectory reaches the current stylus position. Once the noted location is detected again, the LMT subtracts the corresponding frame number for the stylus position from the current frame number to obtain the difference in the frame counts (*frame#*$_{Ct}$ - *frame#* $_{Dt}$). Finally, I take into consideration the frame rate of the camera to output the latency value.

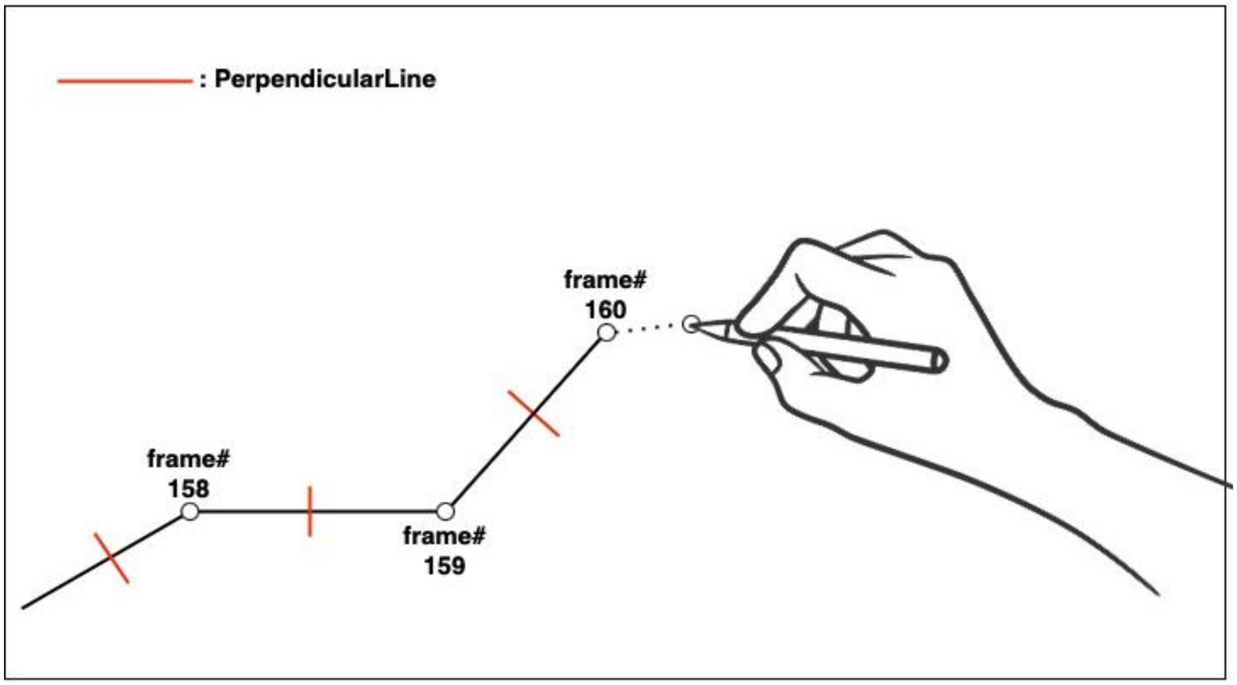
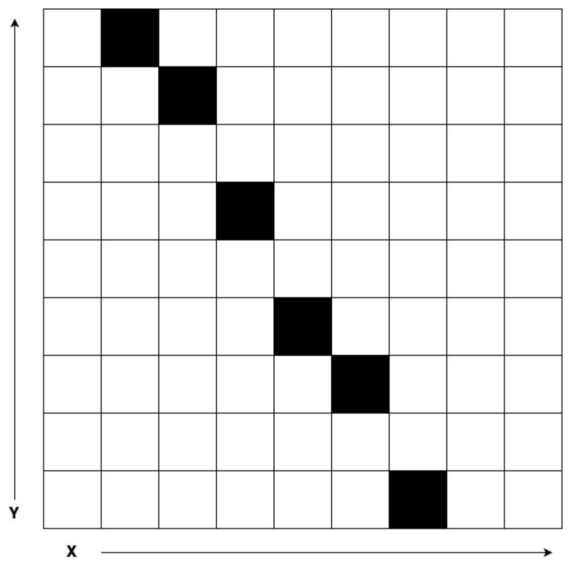
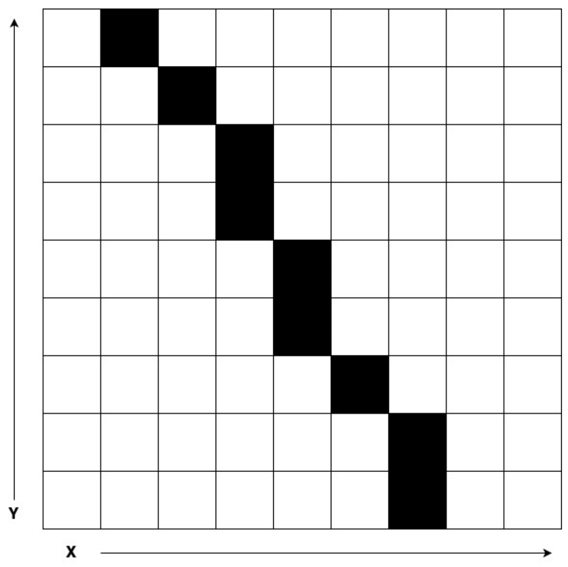


Figure 3: The LMT calculates the perpendicular line based on the slope of the graphic trajectory in each frame. The slope of the perpendicular line determines whether to expand the x-axis or y-axis by 10 units on both sides to expand the users' input area.



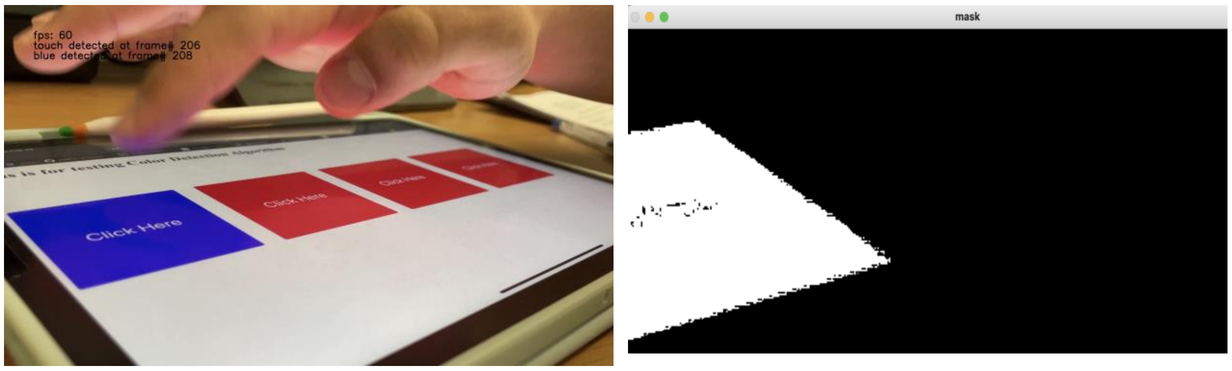


(a) Perpendicular line by expanding x-axis (b) Perpendicular line by expanding y-axis

Figure 4: The results obtained by expanding the user's input area by increasing the x-axis (a) and y-axis (b) when the slope of the perpendicular line is steep. By increasing the y-axis, the chance of having a gap between endpoints can be decreased.

III.2.1 Perpendicular Line Calculation

I utilize a technique known as perpendicular line calculations in order to ensure that the LMT is capable of accurately capturing the resulting output on the screen. This technique allows us to accurately capture the screen output, even when there are variations in the position or orientation of the input device. The LMT first calculates the slope of the perpendicular line of the user-drawn line, as illustrated in Figure 3. Based on the value of the slope, I determine whether to expand the x-axis or y-axis by 10 units on both sides. This process is to prevent the occurrence of any potential gaps or missing segments in the newly drawn line. For instance, when the slope of the newly drawn line is between -1 and 1, the line will be steep. Expanding the x-axis, in this case, would increase the likelihood of the newly drawn line having gaps between endpoints, as shown in Figure 4. Therefore, I instead expand the y-axis when the slope falls within the range of -1 and 1, and expand the x-axis otherwise.



(a) Color changes of buttons when touched

(b) The color mask that captures blue on the screen

Figure 5: Images of the custom color-changing button used to measure the latency of tapping interactions. When the color of the buttons changes to blue(a), the color mask of LMT captures it and notes the corresponding frame number.

III.2.2 Tap Latency Measure

As shown in Table 1, most systems support only one interaction type: either tapping or dragging. To support both tapping and dragging interactions, I adopted a color detection algorithm to measure tap latency with the LMT. I created a simple application that shows color-changing buttons on a screen, initially in red and changing to blue when touched. A mirror placed beside the device can help the detection of the stylus or finger as it touches the screen surface. To measure tap latency, the LMT displays the recorded video frame by frame, prompting the user to click on a keyboard when the moment of tapping is identified to log the corresponding frame number. Then the LMT will automatically capture when the color of the button is changed to blue to log the corresponding frame number and measure the latency using the frame-counting technique.

CHAPTER IV

Experiment

The primary objective of this experiment is to validate the LMT as a case study measuring the latency of two real-world applications on one touch-screen device. The goal is not to provide novel findings about the performance impact of the latency for these particular applications but serves as a usability study for the LMT itself.

I conducted an experiment that is similar to two previous studies (Jota et al., 2013; Ng et al., 2012) but using the LMT for measuring latency. The direct-touch interface latency is a significant factor in user performance degradation. According to Ng et al. (2012), user performance increases as the latency becomes lower, and users can perceive touchscreen latency as low as 2.38 ms. Another study showed that users may be able to perceive latency as low as 1 ms (Jota et al., 2013).

IV.1 Apparatus

The experiment used two drawing applications with different latencies on a tablet device, iPad Air, to test human performance changes. An external camera, an iPhone 12 mini, was fixed on a mount to record the target screen and participant's stylus movement from a top view at 60 frames per second. To initiate the task, two dots with the same distance were presented on both applications, and participants were asked to connect them by drawing a line using the stylus provided. No mirror was used for the tapping experiment because the video recorded from the side was sufficient to capture the moment of touch and button color change. Therefore, the camera was moved to the side of the tablet when the participants were performing the tapping task.

IV.2 Participants

A total of 53 students from Charleston Southern University with prior experience using smartphones or tablets participated in the experiment. All the participants were Computer Science majors and the participation was voluntary without any compensation. Before contributing to the study, participants were informed of the experiment's purpose and the required tasks through a consent form. The form was provided to ensure that participants understood the objectives of the study and were willing to contribute to it.

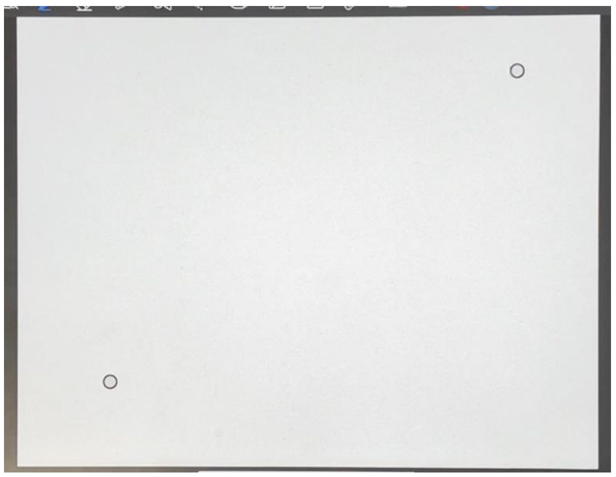
However, the analysis for the drawing task included data from 26 participants, while the analysis for the tapping task included data from only eight out of 53 participants. This was due to certain limitations of the LMT, such as poor lighting conditions and cases where the stylus did not fall within one of the four coordinates of the bounding box determined by the object-tracking algorithm.

	Choices				
Gender	**Male**	**Female**	**Prefer Not To Answer**	**Other**	
Counts	30	23	0	0	
Age	**17 - 19**	**20 - 25**	**26 - 30**	**31 - 35**	
Counts	24	23	4	2	
How much time do you spend online?	**1-2 hours/day**	**3-4 hours/day**	**5-6 hours/day**	**7-8 hours/day**	**More than 8 hours/day**
Counts	2	16	19	3	13
What kind of device do you usually use?	**Smartphone**	**Laptop**	**Desktop**	**Tablet**	
Counts	41	21	8	6	

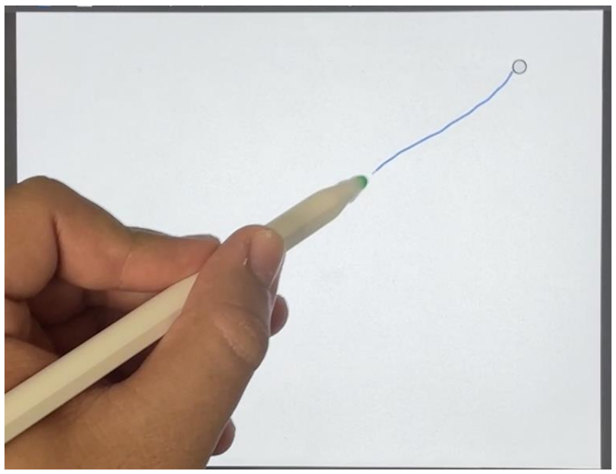
Table 2: Summary of Demographic information

IV.3 Task Procedure

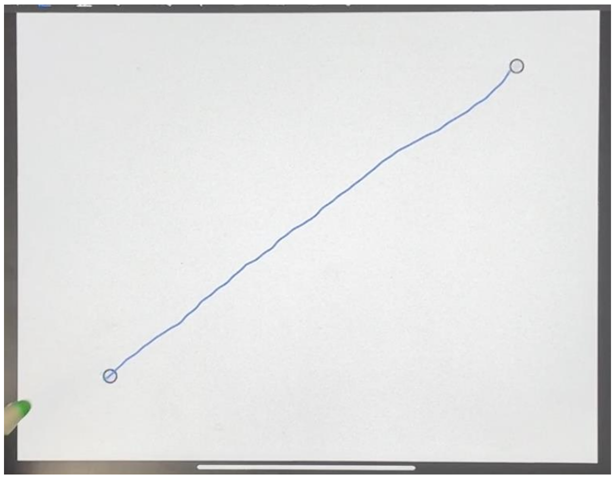
The participants were instructed to come to the desk where the setup for the camera and the touch-screen device was already prepared. As shown in Figure 6, two dots were presented on the application to initiate the task. Then the participants were asked to draw a straight line to connect the two dots on the screen that were 150 mm apart from each other. The dots were presented with a diameter of 5 mm. This task was repeated for the second drawing application. Additionally, the participants were asked to click the color-changing buttons on the screen, shown in Figure 7, which is the custom application used to measure tap latency. The participants' stylus movement and the screen output were recorded simultaneously. Once the participants finished the task, I asked them to answer a short questionnaire about their experience of the study.



(a) The initial screen showing the two dots

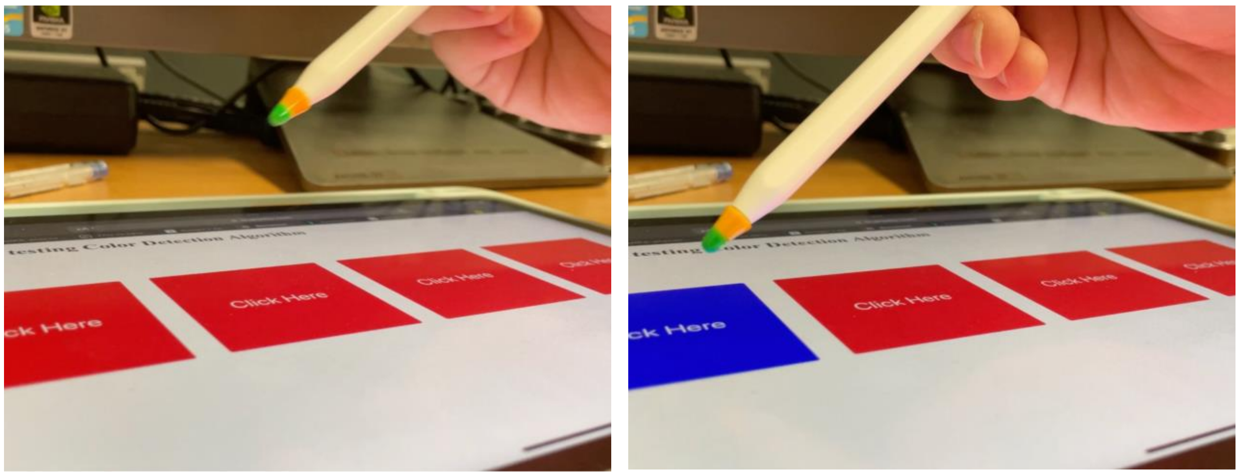


(b) A line drawn to connect the two dots



(c) The completion of the task

Figure 6: Images of the drawing task performed in the experiment. Initially, two dots were displayed on the application to prompt the drawing task (a). Subsequently, the participants were instructed to draw a line connecting the dots, starting from one of them (b). The task was completed once the line reached the other dot (c).



(a) Before tapped (b) After tapped

Figure 7: Images of the tapping task performed the experiment. The task involved color-changing buttons that were initially red (a). When a button was tapped, its color changed to blue (b). The participants were prompted to tap all four buttons.

IV.4 Analysis

The human performance and latency analysis for each drawing application involved video recordings of 26 participants. To analyze latencies in the drawing task, I first cleaned the data to remove any outliers or irrelevant data, such as periods of waiting before or after completing the task. The cleaned data was then used to calculate the latencies of the two drawing applications. The latencies were calculated using the distance-measuring and frame-counting techniques. To analyze latencies in the tapping task, I used video recordings of the tapping task from eight participants and calculated the latencies using the frame-counting technique. Additionally, I conducted a manual analysis to verify the accuracy of the LMT-generated latency results by comparing them with manually measured latencies.

IV.4.1 Drawing Latency Analysis

To accurately assess the latencies of the two drawing applications, I used the latency data obtained from the LMT. However, to avoid including any irrelevant data, such as the period when participants waited before starting or lingered after completing the task, I considered only the middle third of frames of each video recording. For instance, when the whole recording was 300 frames, only the middle 100 frames were used to determine the latency.

The LMT generated two sets of latency data for the two applications, obtained through the distance-measuring and frame-counting techniques. I computed the average latency values for each participant using the values obtained from both techniques. Subsequently, I calculated the average latency values across the 26 video recordings for each application and each technique separately. This allowed me to obtain the average latency values for each application for each technique, providing a comprehensive assessment of the latencies of the two drawing applications.

To verify the accuracy of the LMT-generated latency results, I conducted a manual measurement of latency with one of the drawing applications using an external camera and a ruler. By using a ruler, I could confirm the physical distance the stylus traveled in a certain time period and the physical distance between the stylus and the graphic trajectory. Using the frame rate of the camera, I could draw the latency values manually. The manual results then were compared to the LMT-generated latency results to confirm the accuracy of the LMT.

IV.4.2 Tap Latency Analysis

To analyze the tap latency, I used the LMT with video recordings from eight participants. I clicked a keyboard when I detected a touch on the screen and the LMT automatically detected the moments the button color changed and calculated the time taken for the color-changing buttons on the screen to register the participant's touch. To validate the accuracy of the LMT, I manually calculated the latencies by noting the frame numbers for each input and output, and compared them with the latency values generated by the LMT.

IV.4.3 Human Performance Analysis

I used task completion time and touch-screen device latencies to assess human performance change with different latency values. Task completion time was measured from the moment a participant started drawing a line to the moment the participant removed the stylus from the screen. This approach was to account for participants' waiting time at the end of the task, especially when the latency value is high. Additionally, participants' answers to the questionnaire were taken into consideration to gain insights into their perceptions of different latencies.

IV.5 Hypotheses

Because the video recordings can result in the accuracy of the LMT and changes in human performance, I set hypotheses for two sections: the LMT (T) and the experiment (E).

H_T1: The LMT will draw accurate latency measurements with an offset of fewer than ± 2 ms.

The object-tracking algorithm performs noise detection to capture the correct edges of target objects. Because the algorithm detects even delicate moves of the target objects, I assumed each video frame would contain the changes in object moves. To test the accuracy, I manually measured the latency to compare the result with the output from the LMT.

H_E1: Task completion time will increase as latency increases.

I aimed to study whether users react to the latency or ignore the latent feedback to analyze the effect of latency on human performance. I assumed users would slow down the stylus to compensate for the latency they recognize. Therefore, H_E1 is derived.

H_E2: Users will recognize the device with low latency.

The questionnaire included a question, “Which device do you think reacted faster?”. I aimed to compare the answers to this question to the latency of the two different drawing applications used to confirm this hypothesis.

H_E3: Human performance will decrease as latency increases.

Task completion time and the device's latency can be used to confirm this hypothesis. I assumed the task completion time would increase as latency increases. This comparison can be a measure to analyze the change in human performance.

CHAPTER V

Experimental Results

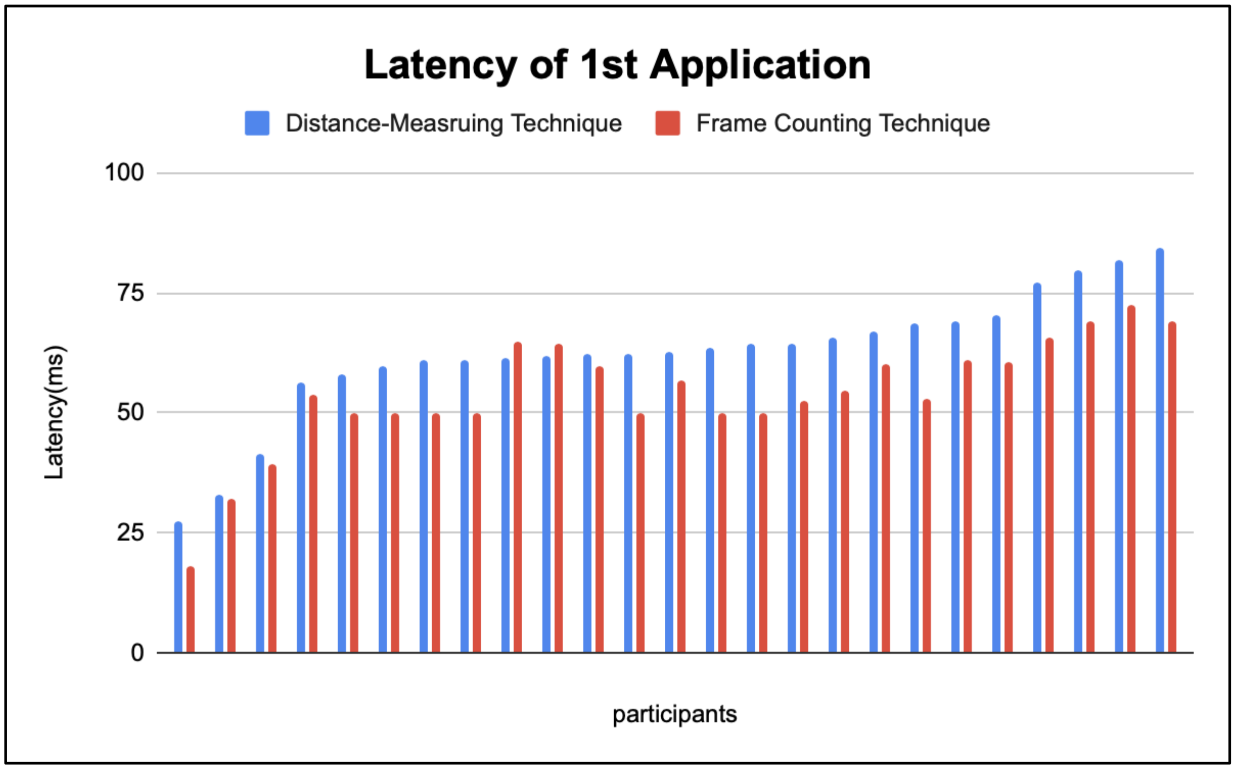
In this chapter, I present the results of the experiment and the manual latency analysis with one of the drawing applications used in the experiment. I used video recordings from 26 out of 53 participants to conduct both human performance and latency analysis, while the tap latency analysis was based on eight video recordings out of 53. The remaining data were excluded from the analysis due to bad light conditions that hindered the use of the LMT, the stylus covering the graphic trajectory due to excessively slow movement, and a few instances where the stylus did not fall within one of the four coordinates of the bounding box determined by the object-tracking algorithm.

V.1 LMT-Generated Latency Values

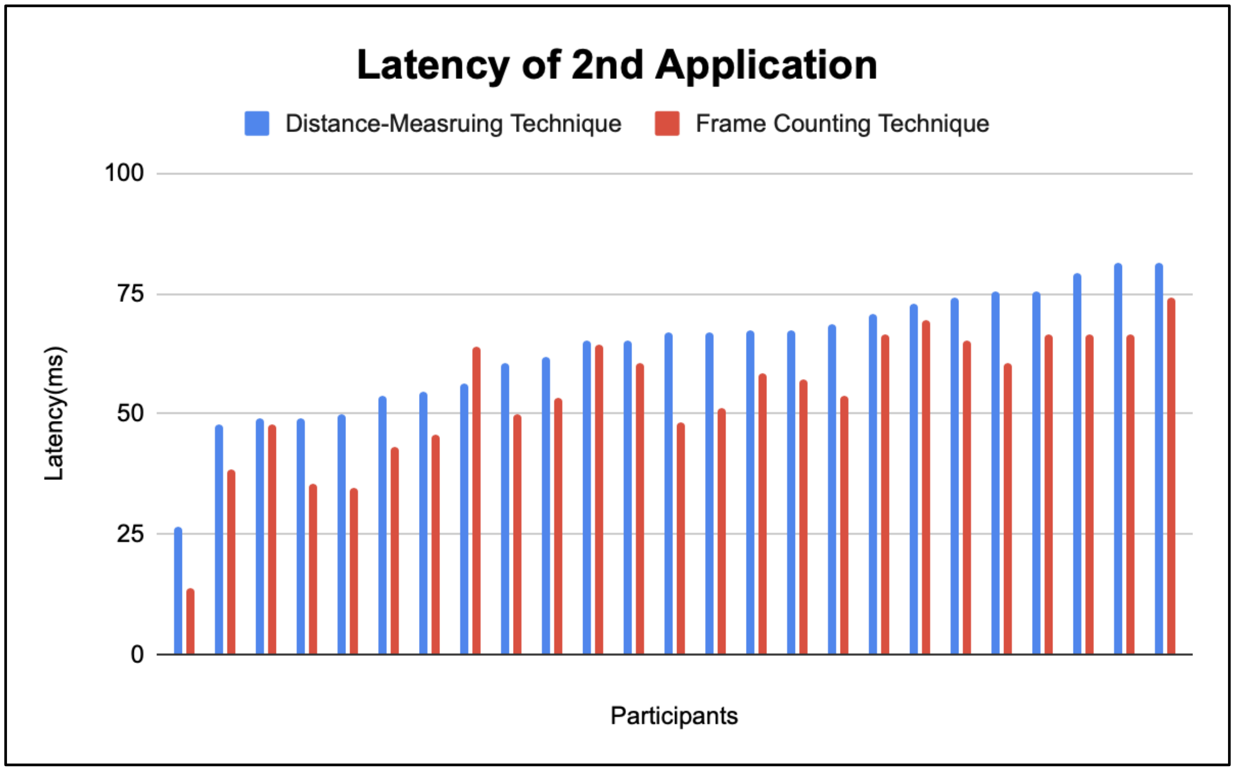
The purpose of measuring the latencies of the drawing applications was to study changes in human performance and to validate the accuracy of the LMT. The LMT generated two sets of latency data for the drawing applications using both the distance-measuring and frame-counting techniques, as these applications involve dragging tasks. The LMT generated latency data for the color-changing button application only used the frame-counting technique, as this application involves tapping tasks.

V.1.1 Latencies of the Drawing Applications

To obtain the overall latency values for each participant, I calculated the average latency values using the data obtained per frame. The resulting average latency values per participant obtained from each technique are presented in Figure 8.



(a) A graph of latency values of the first drawing application arranged from least to greatest.



(b) A graph of latency values of the second drawing application arranged from least to greatest.

Figure 8: A graph of latency values of the two drawing applications. The blue lines depict the latency values achieved from the distance-measuring technique while the red lines depict the latency values achieved from the frame-counting technique.

Figure 8 (a) shows the latency values of the first drawing application obtained from the distance-measuring technique range from 27.42 ms to 84.59 ms, with an average of 62.60 ms while the values from the frame-counting technique range from 17.90 ms to 72.52 ms, with an average of 54.26 ms. The latency values of the second application, (b) in Figure 8, drawn from the distance-measuring technique range from 26.62 ms to 81.58 ms, with an average of 63.57 ms while the values from the frame-counting technique range from 13.89 ms to 74.35 ms, with an average of 54.28 ms.

As shown in Figure 8, the distance-measuring technique yielded slightly higher values on average than the frame-counting technique. To determine the relationship between the data obtained from the two techniques, correlation coefficients, t-score, and p-value were calculated for each application. For the first application, the correlation coefficient between the two techniques was $r = 0.92$, $p < 0.05$, while for the second application, it was $r = 0.91$, $p < 0.05$.

To investigate whether there were significant differences in the mean latency values between the two applications, I performed a t-test using the latency values obtained from each technique. For the distance-measuring technique, t-score was $t = - 0.26$, $p > 0.05$, while it was $t = - 0.005$, p > 0.05 for the frame-counting technique. These findings suggest that there was no significant difference between the means of the two applications, and that detecting any difference would require a larger sample size.

V.1.2 Latency of the Color-Changing Button Application

Because the color-changing button application consists of four buttons, I could obtain four latency values for each participant. The box plots presented in Figure 9 aid in examining the consistency of latency values across participants.

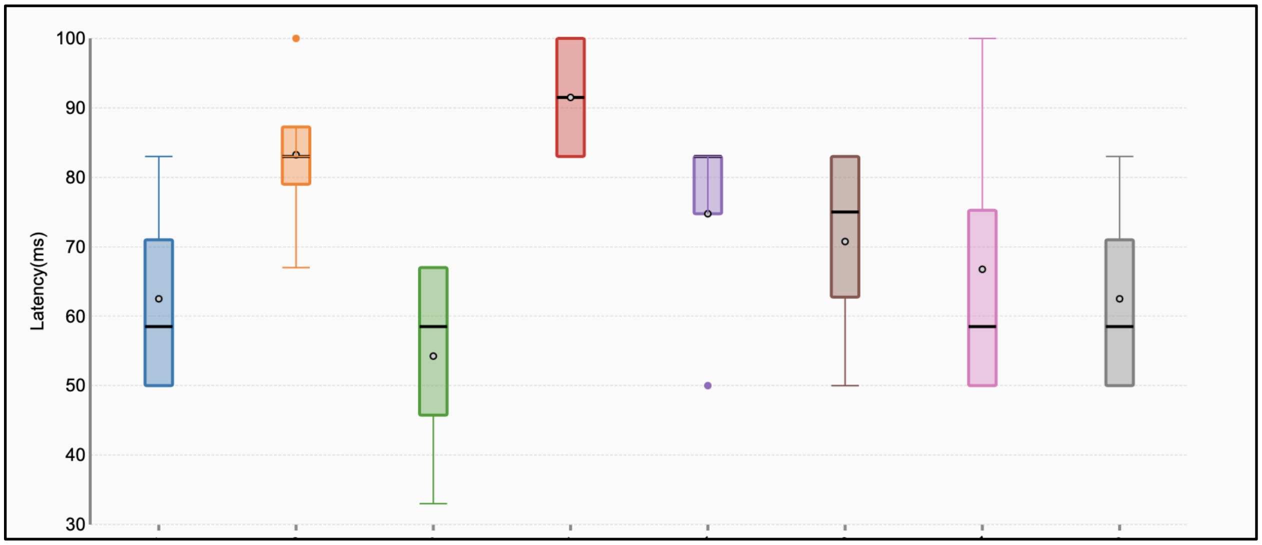


Figure 9: A box plot of latency values of the color-changing button application per participant. The median and average of each data set are depicted by the bold line and dot within the box, respectively.

The average latency values per participant ranged from 54.25 ms to 91.5 ms, with an overall average of 70.78 ms. When comparing the average latency values per participant to the overall average latency, the closest average latency value had a difference of 0.03 ms, while the furthest average latency value had a difference of 20.71 ms. The average latency values of five participants varied by less than 10 ms from the overall average latency, while the remaining three participants had latency values differing by more than 10 ms but less than 20 ms from the overall average.

V.2 Latency Values From Manual Analysis

To assess the precision of the LMT, I manually measured the latency of the first drawing application. I performed the same drawing task that was executed in the experiment. I ran the LMT three times using the same video recording and conducted the manual analysis once.

Because the LMT provided consistent latency values across all three runs, Figure 10 shows only one line for the latency values achieved from the LMT.

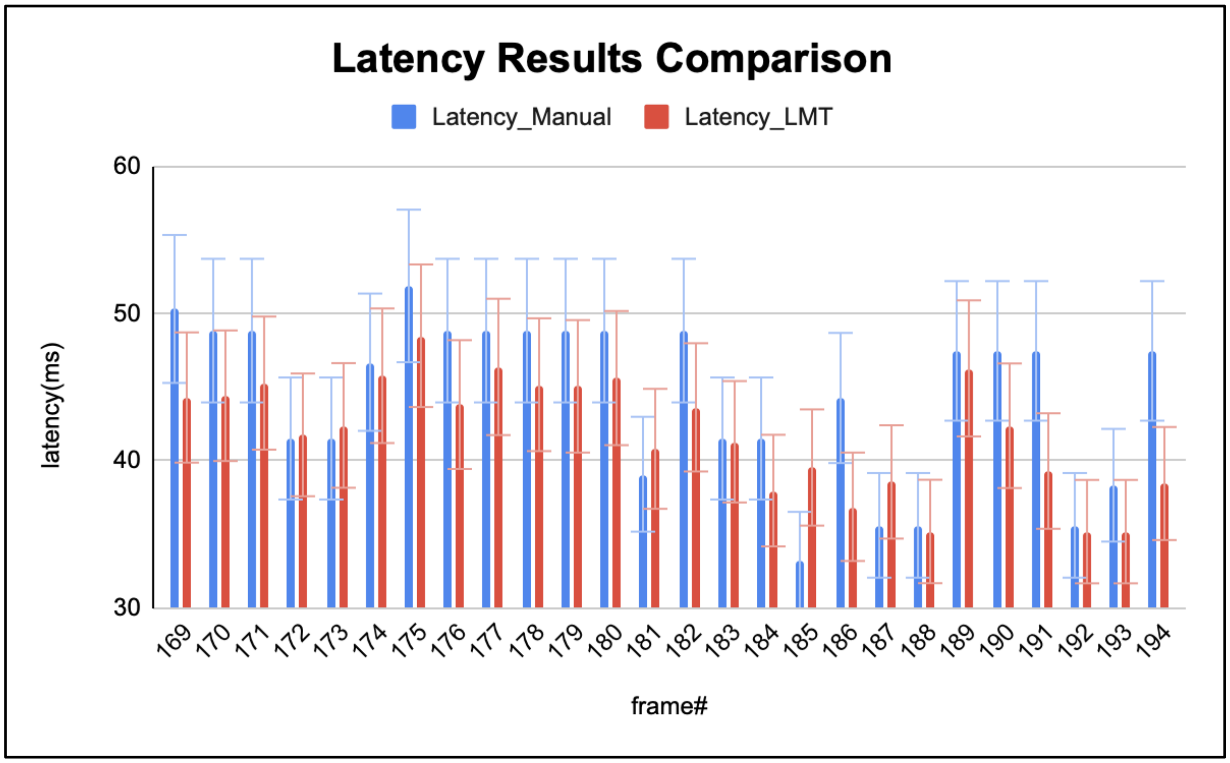


Figure 10: A graph of latency values from manual analysis and the LMT, both using the same video recording. The blue line depicts the latency values achieved from the manual analysis while the red line depicts the latency values achieved from the LMT.

Method	**Mean Latency(ms)**	**Variance(ms^2)**	**Standard Deviation(ms)**	**Standard Error(ms)**
Manual	44.49307085	29.90882313	5.46889597	1.072538741
LMT	41.86923077	15.04527138	3.878823454	0.7606998646

Table 3: Comparison of latency values obtained from the LMT and manual analysis.

As shown in Figure 10, the latency values drawn from the LMT range from 35.16 ms to 46.36 ms, with an average of 43.58 ms while the values from the manual analysis range from 33.18 ms to 51.86 ms, with an average of 44.04 ms. To determine the relationship between the data obtained from the two techniques, correlation coefficients, t-score, and p-value were calculated. The correlation coefficient was $r = 0.77$, $p < 0.05$. Furthermore, t-score for the data sets was t = 1.99, p < 0.05. The effect size (Cohen's d) for this two-sample t-test was $d = 0.567$, which is a medium effect size. Cohen suggested that the effect size $d = 0.2$ is considered a small effect size, 0.5 represents a medium effect size, and 0.8 a large effect size. This guideline suggests that when the effect size is small, the difference is negligible, even if it is statistically significant.

The correlation coefficient value indicates that there is a strong positive linear relationship between the LMT and manual analysis values, indicating that the LMT produces reliable latency values. However, there is still a difference between the means of the two sets of values, which means that the LMT produces latency values that are slightly lower than the manual analysis.

V.3 Human Performance Evaluation Result

To evaluate the impact of latency values on human performance, I analyzed the latency values for each application and compared them with the task completion time to find a relationship. Table 4 and 5 present the latency values obtained from the two techniques and the task completion time for the first and second drawing applications, respectively.

Participant#	**Latency_Distance Measuring Technique(ms)**	**Latency_Frame Counting Technique(ms)**	**Task Completion Time(frame#)**
2	62.2875	59.52571429	44
4	79.51456522	68.96793103	137
7	59.73970588	50	103
8	61.07261905	50	130
10	57.94969697	50	96
20	60.93758621	50	89
21	63.3525	50	113
22	62.3225	50	119
24	64.6	52.56461538	99
26	41.63588235	39.28571429	108
30	77.068	65.87142857	138
34	27.42339623	17.90259259	163
36	56.499375	53.57142857	103
38	70.30870968	60.41875	94
39	64.24142857	50	91
42	61.69333333	64.28857143	64
43	68.66866667	53.125625	93
44	61.50434783	64.81777778	71
45	67.02851852	60.002	84
46	84.59236842	69.166	120
47	81.82222222	72.52216216	173
48	65.71754717	54.7625	168
49	33.04153846	31.94166667	81
50	69.22777778	61.11333333	111
52	62.6283871	56.66733333	104
Correlation Coefficient	**0.1375457742**	**-0.05657492891**	
P-Value	**0.0000001448002699**	**0.00000002170203354**	

Table 4: Data of latency values and task completion time measured per participant for the first drawing application.

Participant#	**Latency_Distance-Measuring Technique(ms)**	**Latency from Frame Counting Technique(ms)**	**Task Completion Time(frame#)**
2	56.43166667	63.89166667	38
4	67.195	58.335	108
7	60.5116129	50	94
8	75.40666667	66.67	71
10	67.41571429	57.14428571	126
20	61.94473684	53.50947368	113
21	53.88119048	43.332	127
22	66.85606061	48.14777778	120
24	49.11666667	47.91625	122
26	49.12682927	35.29117647	122
30	47.944375	38.45923077	97
34	26.62603774	13.89055556	157
36	65.281875	64.58625	95
38	67.09883721	51.19071429	131
39	68.74241379	53.84692308	89
42	74.30045455	65.38307692	66
43	81.58515152	74.35692308	102
44	73.08086957	69.44666667	72
45	70.78657143	66.67	105
46	65.43219512	60.41875	123
47	54.71821429	45.61315789	167
48	50.01380952	34.84545455	138
49	81.31617647	66.67	103
50	75.2955814	60.71642857	93
52	79.17185185	66.67	80
Correlation Coefficient	**-0.5165495861**	**-0.6618618377**	
P-Value	**0.000003069405029**	**0.0000003875782428**	

Table 5: Data of latency values and task completion time measured per participant for the second drawing application.

To find a relationship between the latency values and task completion time, I calculated correlation coefficients and p-values for each application and technique combination. For the latency data obtained from the distance-measuring technique in the first drawing application, the correlation coefficient was $r = 0.13$ with $p < 0.001$, while for the latency data obtained from the frame-counting technique, was $r = -0.05$ with $p < 0.001$. For the second application, the latency data obtained from the distance-measuring technique showed a correlation coefficient $r = -0.51$ with $p < 0.001$, while the latency data obtained from the frame-counting technique showed $r = -0.66$ with $p < 0.001$.

V.4 Questionnaire Results

To answer the hypothesis H_E2, I compared the latency values of the drawing applications and the participants' answers to the questionnaire. The results from the questionnaire showed that the majority of the participants spent at least 3-4 hours per day online and used smartphones and laptops as their primary electronic devices. Half of the participants chose the first application as the one with faster response time, while the rest selected the second. The majority of participants rated the response time of both applications as 'Neutral' or 'Noticeably Fast'. However, 10 participants who selected the first application as faster rated the second application as 'Noticeably slow'. In terms of ease of use, most participants found both applications to be very easy or a little easy. Some participants reported needing to slow down the stylus movement or keep the speed of the stylus movement, while the most common response was that they did not need to compensate for any delay.

	Choices				
Which application do you think reacted faster?	**First App**		**Second App**		
Counts		26			26
How fast or slow would you rate the response time of the first application to your interactions?	**Very Slow**	**Noticeably Slow**	**Neutral**	**Noticeably Fast**	**Very Fast**
Counts	0	2	22	22	7
How fast or slow would you rate the response time of the second application to your interactions?	**Very Slow**	**Noticeably Slow**	**Neutral**	**Noticeably Fast**	**Very Fast**
Counts	0	10	14	22	7
How difficult was it to use the first application considering its response time?	**Very Difficult**	**A Little Difficult**	**Neutral**	**A little Easy**	**Very Easy**
Counts	0	5	16	8	24
How difficult was it to use the second application considering its response time?	**Very Difficult**	**A Little Difficult**	**Neutral**	**A little Easy**	**Very Easy**
Counts	0	5	13	17	18
Did you need to compensate for the delay in the applications' response to your interactions? If so, how?	**I did not need to compensate for the delay**	**Slowed down the stylus movement**	**Kept the speed of the stylus constant**	**Moved the stylus faster**	**Other**
Counts	22	14	14	2	1

Table 6: Questionnaire summary table. Counts represent the number of participants who chose each option.

V.5 Usability of the LMT

During the analysis, the LMT proved to be highly useful in the experiment, providing convenience and speed in measuring latency. Compared to manual measurement, which took one minute per frame, the LMT took just a few seconds to obtain latency values per video recording.

The frame-by-frame display of video recording showing the location of the stylus and graphic trajectory also made it convenient to observe the physical distance between them. Another advantage was the color mask window that helped in ensuring the correct screen output moment was captured when measuring latency with the tapping task. However, the LMT's limitation in supporting calibration in converting pixel measurements to physical distances caused an inconvenience. Specifically, recording the target screen from a top-view video using a camera mount and aligning the target screen with the camera grid required extra care. Another challenge faced during the latency analysis was the exclusion of some recordings due to poor lighting conditions that disrupted the color detection algorithm for capturing the blue graphic trajectory on the screen.

CHAPTER VI

Discussion

The findings of the results chapter are interpreted in this chapter and recommendations for researchers studying latency are provided. Additionally, I discuss the inherent and LMT-specific limitations that impact the generalizability, validity, reliability, and robustness of the latency measurement.

VI.1 Accuracy of the LMT

The results from the experiment showed that the LMT is reliable and provides accurate latency measurements. Although the distance-measuring technique yielded slightly higher values on average than the frame-counting technique, when measuring latency with drawing tasks, both techniques were found to be valid and reliable, as evidenced by the strong correlation between the two techniques. The strong correlation values suggest that the differences in values obtained from the two techniques may be attributed to the inherent differences in the measurement methods. Furthermore, the strong positive correlation between the LMT-generated and manually measured latency values indicates that the LMT is accurately measuring latency and producing consistent results. The fact that the LMT-generated latency values were the same when the LMT was run three times on the same video recording further supports the reliability of the LMT. However, there was an 18 ms difference in the average latency values obtained from the 26 participant data compared to the manual analysis. This difference may be attributed to factors such as limitations of camera frame rate or differences in underlying operating system conditions during the manual analysis. For example, if the stylus movement was much faster in the manual analysis because the camera can capture only 60 frames per second, the LMT might result in less latency values than the actual latency. Additionally, because the manual analysis and the experiment took place at different times, the application may have shown different latency values due to different underlying operating system conditions.

The results from the experiment with the tapping task also demonstrate that the LMT provides accurate latency measurements. The average latency values obtained from the participants were consistent and the differences between the individual participant's average latency values and the overall average were minimal, with the closest average latency value differing by only 0.03 ms. Even participants whose latency values varied by more than 10 ms from the overall average still demonstrated a high degree of consistency. These findings suggest that the LMT is reliable and produces accurate latency measurements.

Overall, based on the results from the accuracy validation of the LMT, the hypothesis H_T1 is supported that the LMT will draw accurate latency measurements with an offset of fewer than ±2 ms .

VI.2 Impact of Latency on Human Performance

When comparing the latency values with task completion time for each participant, I could not find a significant relationship between the two values. Even though the first drawing application showed slightly faster latency, it did not impact human performance accordingly which leads to the conclusion that the hypothesis H_E1 and H_E3 are not supported. One of the reasons was the simplicity and short duration of the task, which made it difficult to detect changes in human performance. Additionally, the small difference in latencies between the two applications did not significantly impact human performance. In the questionnaire, half of the participants chose the first application as the one with faster response time, while the rest selected the second. Nonetheless, the majority of participants rated the response time of both applications 'Noticeably Fast', indicating their inability to discern a latency difference between the two. Therefore, H_E2 is not supported.

It is important to note that the lack of a significant effect does not invalidate the experiment, but suggests that further research is necessary. Therefore, I believe that future work should consider using more complex and longer tasks that can challenge the participants or better simulate real-life scenarios to obtain meaningful data. Additionally, the study can be further improved by recruiting more participants with various levels of experience with smartphones or tablets, as well as using a wider range of latency values to capture a more significant impact.

VI.3 Limitations and Future Improvements

While the LMT has shown promising results in accurately measuring the touchscreen latency, there is still much room for improvement. This section will illustrate each aspect that could improve the performance and usability of the LMT. By recognizing the limitations of the study, we can avoid overgeneralization or misinterpretation of the results and improve future research.

VI.3.1 Frame Rate

The frame rate of the camera used to record the screen output plays a critical role in the accuracy of the latency measurements. Higher frame rates allow for more delicate capturing of the movement of the stylus and graphic trajectory. In my experiment for human performance, the videos were recorded at 60fps. However, a higher frame rate of 120fps or more would provide twice the number of coordinates for the stylus and graphic trajectory respectively, leading to more precise latency measurements. Therefore, I recommend that future studies explore the use of cameras with higher frame rates to improve the precision of the LMT.

VI.3.2 Video Image Resolution

The image resolution of the camera used to record the screen output is also important for precise latency measurements. The LMT relies on precise coordinates of the stylus movements and the graphic trajectory in each frame to determine the velocity and distance between them. Hence, the higher the pixel resolution of the camera, the more detailed and precise the coordinates will be, resulting in more accurate latency values. For instance, consider a scenario where the same video is recorded using a camera with a resolution of 640 x 480 pixels and another camera with a resolution of 1920 x 1080 pixels. The latter camera will have three times as many pixels in the same physical space, resulting in more delicate and precise coordinates for the objects in the video. As a result, using a higher pixel resolution camera can lead to more accurate measurements of latency. Ideally, the recorded video will have at least twice the resolution of the device for which the latency is being measured.

VI.3.3 Distance Measurement

When measuring the distance between the current position of the stylus and the graphic trajectory, the LMT uses the Euclidean distance between the two points. While this method can result in accurate distance when the drawn line is perfectly straight, it may not be accurate when the line is curved or not perfectly straight. Many systems assume that the line is straight when using the distance-measuring technique (Bérard & Blanch, 2013; Ng et al., 2012; Yun et al., 2017). However, given the importance of distance in measuring latency, it is worth considering alternative approaches that can improve accuracy.

One possible method is to use the physical length of the drawn line in the previous frame, instead of the distance between the stylus and the graphic trajectory in the current frame, to calculate distance. This approach captures the physical length of the drawing that is being delayed, resulting in more accurate latency measurements, especially when the drawn line is not perfectly straight. Additionally, calculating the length of the line using the arc length integration, which involves breaking the line into small segments and computing the length of each segment using calculus, can result in a more accurate measurement of the length of the drawn line. However, this approach is more computationally intensive than using Euclidean distance.

VI.3.4 Color Detection Algorithm

The LMT uses a color detection algorithm to distinguish between the stylus and graphic trajectory. The LMT uses two distinct ranges of masks, one for detecting green and one for detecting blue. However, the requirement to use specific colors for the stylus and graphic trajectory could be seen as a constraint for some evaluators. Additionally, the lighting conditions of the room where the video is taken can also impact the algorithm's effectiveness. For example, during the human performance experiment, eight videos out of 53 had to be excluded due to poor lighting conditions that disrupted the video's recording of the blue graphic trajectory. In the case of the tapping task, the blue light from the screen disrupted the color detection algorithm, rendering the majority of video recordings unusable. Moreover, the blue light appeared more prominently in the camera when the screen was recorded from the side. However, despite this limitation, using different ranges of masks could still be considered flexible because evaluators can adjust color mask ranges to use different colors or compensate for lighting disturbances. Of course, making such adjustments might require some level of technical expertise or trial and error to achieve optimal results. Furthermore, the use of colored caps on the stylus tip could impact the participant’s natural drawing experience. Therefore, future work could focus on developing a more robust and flexible color detection algorithm that can perform well under different lighting conditions and allow for more natural user interaction without the need for colored caps.

VI.3.5 Transform the Digital Shape into the Physical Dimension

To accurately measure distances in a video, the LMT requires calibration to convert pixel measurements to physical distances. However, currently, the LMT only supports calibration when the video is taken from the top view. Enabling geometric transformation in the LMT to transform the shape of the target screen to the physical dimension will improve the usability of the LMT, allowing evaluators to take videos from any angle without worrying about calibration. This will enhance the LMT's flexibility and ease of use, making it accessible to a wider range of users. Additionally, enabling the transformation will lead to more accurate distance results when using the distance-measuring technique, resulting in more precise latency measurements. Future work could focus on developing an algorithm that can automatically transform the shape of the target screen to its physical dimensions, eliminating the need for manual camera angle adjustment.

VI.3.6 Automatic Target Screen Detection

When using the LMT, calibration is required to convert pixel measurements to physical distances. Currently, evaluators need to manually click the four edges of their target screen to perform this calibration. However, the accuracy of distance calculation could be influenced by the accuracy of the evaluator's clicks. While the accuracy of the LMT showed that various user clicks resulted in less than a 1% error, careful click performance is still necessary to achieve accurate calibration. Therefore, future improvements could focus on developing a function that enables an automatic target screen size detection that utilizes contour features. This improvement can eliminate the need for evaluators to manually click the target screen edges, resulting in more accurate and consistent physical distance measurements.

VI.3.7 Improved Stylus Tracking using Custom Shapes

The current object-tracking algorithm used by the LMT calculates the distance between the stylus and graphic trajectory based on the coordinates of the four edges of each bounding box. However, using a square-shaped bounding box could result in inaccuracies when calculating the distance if the stylus tip is not located on one of the edges due to the angle of the stylus. This limitation was observed during the human performance experiment, where four videos out of 53 had to be excluded. Therefore, future improvements could include the use of custom shapes for the bounding box using contour features or considering all coordinates of the bounding box while calculating the distance to eliminate the current problem. However, these improvements could be computationally more expensive, potentially slowing down the LMT. Therefore, a balance between accuracy and speed should be considered while implementing these improvements.

VI.3.8 Generic Application Support

Although the LMT has shown promising results in measuring touchscreen latency, the LMT is currently limited to simple drawing applications where there are no disturbing objects on the target screen. However, extending the functionalities of the LMT to support more generic applications and movements would be beneficial considering its vast use cases. The current approach can be improved by incorporating advanced algorithms for object segmentation, color filtering, and motion tracking to allow the LMT to perform reliably in more complex scenarios. Therefore, future work can focus on developing more sophisticated algorithms and expanding the LMT's capabilities to increase its usability in various applications.

CHAPTER VII

Conclusion

In conclusion, the automated latency measuring tool (LMT) offers an approachable and effective solution to measure the latency of touchscreen devices. The results from the human performance experiment and manual latency analysis indicate that the LMT is reliable and produces accurate latency measurements for both drawing and tapping tasks. However, there are several limitations to the LMT that should be considered in future research. These limitations include the need for transforming the digital shape into the physical dimension and custom shape of the bounding box for the object tracking algorithm, as well as the use of a more versatile color detection algorithm to allow for greater flexibility in detecting touch events and screen output. Furthermore, future studies should consider using more complex and longer tasks to simulate real-world scenarios in studying the impact of latency on human performance changes.

Overall, the LMT presents a novel and accessible method for measuring the latency of touchscreen devices, without the need for extra hardware except an external camera, which can ease the burden on system designers. This study provides valuable insights into the measurement of latency on touchscreen devices and highlights the need for further research in this area to improve the accuracy and reliability of latency measurements. By addressing the limitations identified in this study, future research can provide more robust and accurate tools to measure the latency of touchscreen devices.

References

Bérard, F., & Blanch, R. (2013). Two touch system latency estimators: High accuracy and low overhead. *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, 241–250. https://doi.org/10.1145/2512349.2512796

Bockes, F., Wimmer, R., & Schmid, A. (2018). LagBox—Measuring the Latency of USB-Connected Input Devices. *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–6. https://doi.org/10.1145/3170427.3188632

Deber, J., Araujo, B., Jota, R., Forlines, C., Leigh, D., Sanders, S., & Wigdor, D. (2016). Hammer Time!: A Low-Cost, High Precision, High Accuracy Tool to Measure the Latency of Touchscreen Devices. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2857–2868. https://doi.org/10.1145/2858036.2858394

Jota, R., Ng, A., Dietz, P., & Wigdor, D. (2013). How fast is fast enough?: A study of the effects of latency in direct-touch pointing tasks. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2291–2300. https://doi.org/10.1145/2470654.2481317

Kaaresoja, T., & Brewster, S. (2010). Feedback is... late: Measuring multimodal delays in mobile device touchscreen interaction. *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction on - ICMI-MLMI '10*, 1. https://doi.org/10.1145/1891903.1891907

Ng, A., Lepinski, J., Wigdor, D., Sanders, S., & Dietz, P. (2012). Designing for low-latency direct-touch input. *Proceedings of the 25th Annual ACM Symposium on User Interface*

Software and Technology - UIST '12, 453. https://doi.org/10.1145/2380116.2380174

Pavlovych, A., & Stuerzlinger, W. (2009). The tradeoff between spatial jitter and latency in pointing tasks. *Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems - EICS '09*, 187. https://doi.org/10.1145/1570433.1570469

Steed, A. (2008). A simple method for estimating the latency of interactive, real-time graphics simulations. *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology - VRST '08*, 123. https://doi.org/10.1145/1450579.1450606

Teather, R. J., Pavlovych, A., Stuerzlinger, W., & MacKenzie, I. S. (2009). Effects of tracking technology, latency, and spatial jitter on object movement. *2009 IEEE Symposium on 3D User Interfaces*, 43–50. https://doi.org/10.1109/3DUI.2009.4811204

Vrouwenvelder, S., Postema, F., & Pool, D. M. (2021, January 11). Measuring the Drag Latency of Touchscreen Displays for Human-in-the-Loop Simulator Experiments. *AIAA Scitech 2021 Forum*. AIAA Scitech 2021 Forum, VIRTUAL EVENT. https://doi.org/10.2514/6.2021-0896

Yun, M. H., He, S., & Zhong, L. (2017). Reducing Latency by Eliminating Synchrony. *Proceedings of the 26th International Conference on World Wide Web*, 331–340. https://doi.org/10.1145/3038912.3052557