Lee, Injung; Kim, Sunjun; Lee, Byungjoo

Geometrically compensating effect of end-to-end latency in moving-target selection games

Published in:
CHI 2019 - Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems

DOI:
10.1145/3290605.3300790

Published: 02/05/2019

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Lee, I., Kim, S., & Lee, B. (2019). Geometrically compensating effect of end-to-end latency in moving-target selection games. In *CHI 2019 - Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* Article 560 ACM. https://doi.org/10.1145/3290605.3300790

Geometrically Compensating Effect of End-to-End Latency in Moving-Target Selection Games

Injung Lee
KAIST, Republic of Korea
edndn@kaist.ac.kr

Sunjun Kim
Aalto University, Finland
sunjun.kim@aalto.fi

Byungjoo Lee*
KAIST, Republic of Korea
byungjoo.lee@kaist.ac.kr

ABSTRACT

Effects of unintended latency on gamer performance have been reported. End-to-end latency can be corrected by post-input manipulation of activation times, but this gives the player unnatural gameplay experience. For moving-target selection games such as Flappy Bird, the paper presents a predictive model of latency on error rate and a novel compensation method for the latency effects by adjusting the game's geometry design – e.g., by modifying the size of the selection region. Without manipulation of the game clock, this can keep the user's error rate constant even if the end-to-end latency of the system changes. The approach extends the current model of *moving-target selection* with two additional assumptions about the effects of latency: (1) latency reduces players' cue-viewing time and (2) pushes the mean of the input distribution backward. The model and method proposed have been validated through precise experiments.

CCS CONCEPTS

• **Human-centered computing → HCI theory, concepts and models**;

KEYWORDS

Latency compensation; moving-target selection

ACM Reference Format:
Injung Lee, Sunjun Kim, and Byungjoo Lee*. 2019. Geometrically Compensating Effect of End-to-End Latency in Moving-Target Selection Games. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland UK*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3290605.3300790

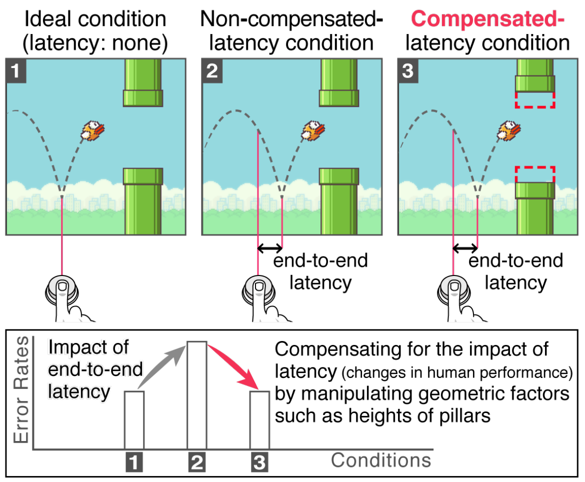

Ideal condition (latency: none)
Non-compensated-latency condition
Compensated-latency condition
1
2
3
end-to-end latency
end-to-end latency
Error Rates
Impact of end-to-end latency
Compensating for the impact of latency (changes in human performance) by manipulating geometric factors such as heights of pillars
Conditions
1
2
3

Figure 1: Geometric compensation overview

1 INTRODUCTION

Minimizing a discordance between user input and system response is crucial to producing seamless human–computer interaction [11, 41]. The temporal gap between the input and the response, which is called *latency*, *lag* or *delay*, is among the factors that create this discordance. The cumulative effect, referred to as *end-to-end latency*, has various sources including device delay, network delay, and processing delay [22]. Latency has an especially strong unintended effect on user experience when accurate time management is critical: in applications such as real-time games. Addressing such situations, this paper has two goals: (1) to provide a clear understanding of how latency changes the user experience and (2) to suggest a novel method of compensating for latency effects.

Latency, which is not intentionally created or considered, has always existed in typical computing environments. The amount of end-to-end latency varies from computing platforms and their conditions, ranging from 21–277 ms for desktop to over 50 ms for touchscreen environments [10, 34, 52]. Latency is often inevitable, not controllable, and vary among devices. According to previous research, the effects of latency on performance are mostly negative [5, 32]. In fact, even near-zero latency may have a negative impact on users, at least

*Corresponding author.

in theory [44]. Also, positive effects of latency on usability have been reported in several recent works [24, 27, 29].

Various studies have been conducted to compensate the unintended effects of latency, especially in *real-time* network games, such as an online first-person shooter (FPS) games. The most widely used method is to rewind the timeframe by the amount of latency and then determine the feedback on the basis of that point [30, 39]. At this point, all game elements, except local input feedback, are rendered in keeping with the rewound game timeframe. Hence, the player experiences a discrepancy between the perceived situation stemming from the local feedback and the feedback given by the compensated system (e.g., an apparent success getting deemed a failure, or vice versa). This can create a serious hindrance to user engagement in real-time games [40].

However, if the latency mechanisms that influence game performance can be clearly explained, *geometric* compensation could be attempted. For example, if 50 ms system latency exists in an FPS game and the player's number of hit points is reduced by 15% relative to a zero-latency situation, an enemy's size can be increased (making it easier to hit) to compensate for the drop in hit rate. For determining the amount of the target's size increase, a predictive model is more effective than tedious trial-and-errors.

To demonstrate the potential of geometric compensation for latency, we focus on one type of task, we call *moving-target selection* task, which is widely encountered with today's mobile games. In this task, players must activate a button within a short time window when a moving target is placed within a certain selection region. The time window is often shorter than typical human reaction times (from 200 to 300 ms depending on the stimulus modality) [17, 18, 45, 50]. Therefore, the player must plan the input in advance for successful trial. Moving-target selection games are popular these days, e.g., Flappy Bird, Dancing Line, etc (see Figure 2). Our survey[1] of the Google Play market revealed that 16 of "the 100 most popular" free games employ a moving-target selection task as their central component.

For a moving-target selection task with zero latency, a model for predicting the user's error rate already exists [28, 29]. We complement the model with the following two assumptions: (1) latency reduces the player's cue-viewing time, and (2) latency pushes the player's implicit aim point backwards. The new model built on these assumptions accurately predicts the effect of end-to-end latency on user error rate for the task. When geometric compensation is applied, the model can predict the corresponding error rate, thereby facilitating control of game difficulty level when latency exists.

Contributions of this research can be summarized thus:

[1] Data retrieval date: September 12, 2018

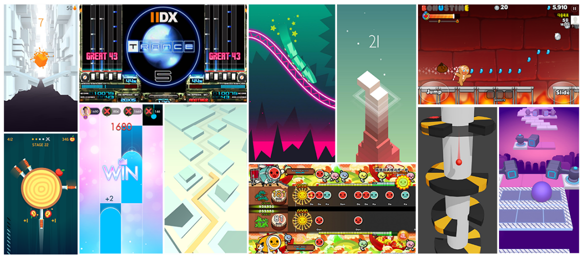


Figure 2: Moving-target selection games: (from upper left) Jelly Jump!, Beatmania, Rollercoaster Dash, Stack, CookieRun for Kakao, Knife Hit, Piano Magic Tiles, Dancing Line, Taiko no Tatsujin, Helix Jump, and Rolling Sky

- We constructed a predictive model that explains the effect of end-to-end latency on user error rate in moving-target selection games.
- We verified the proposed model (R^2=0.94) by using a customized experimental apparatus that can precisely control the latency (<0.5 ms accuracy, <1.5 ms precision).
- We demonstrated that geometric compensation can offset the unintended effects of end-to-end latency.

2 RELATED WORK

Latency and its Measurement

Latency, lag or delay is the time span between a input and its corresponding response [8, 32, 38, 46]. *End-to-end latency* generally refers to the total time elapsed between a user's initiation act and the system's visual, auditory, or other responses [10, 21]. Several components contribute to end-to-end latency in an interactive system. In online mobile games, there are three types of latency: device delays, network delays, and processing delays [22]. For nowadays touchscreen mobile games, device latency (capacitive touch sensor and display) and processing latency (logic in game software) exist as a baseline [34]. Network communication latency may added for online games. All kinds of delays (device, processing, and network) increase user reaction time [41].

Often, measuring end-to-end latency entails using a camera, in most cases a high-speed one, to capture the input motion and the associated feedback [6, 16, 21, 31, 34, 49]. Latency is calculated by manually counting the frames between them. Automated tools such as a mechanical relay [13], optical sensor [13], or accelerometer [8] have been employed to capture the exact input and output moments. The automation allows repeated measure for better precision. Depending on the devices and applications, the amount of end-to-end latency ranges from 21 to 277 ms, composed of input latencies of 2–30 ms and output latencies between 19 ms and 247 ms [10]. Latencies in touchscreen devices such as smartphones or tablet PCs range from 50 to 200 ms [15, 21, 52].

Effects of Latency on Human Performance

Latency influences user experience in interactive systems. Firstly, even a small amount of latency is perceivable. Interactive display delays start to be detectable at 40 ms to 90 ms [41]. According to Ng et al., people have been able to visually detect even latency as low as 2.38 ms when performing direct-touch dragging [34]. Latencies that are greater than "barely perceptible" start to impair user experience [20, 40]. Practically, delays more than 180 ms in FPS game have tended to frustrate users, which lead them to eventually stop playing [3]. Latency of 600 ms for tap actions and 450 ms for drag actions annoyed people using touchscreens [1, 43].

In addition disturbed user interaction, higher latency hinder human performance [2] also. With general tasks such as programming and problem-solving, it has been found that latency impairs work efficiency [4, 33, 46, 48]. For target-acquisition tasks specifically, movement times and error rates have risen as the amount of latency grew [12, 32]. More precisely, latencies over 100 ms have strong negative effects on human performance [12, 32]. FPS game players exposed to 200 ms latency exhibited a performance deduction of 1 kills-per-minute[2] compared to players experiencing 45 ms latency [3]. Human performance degrades more with targets that are smaller or further away [19].

Effects of Latency on Anticipated-Input Tasks

Obviously, latency negatively affects users' performance of reaction tasks (responding as quickly as possible to a given unexpected stimulus). However, in real-time games, there are often anticipation tasks: a user can anticipate the moment of input before the actual stimulus. Imagine a target that briefly blinks periodically or that moves toward a particular selection region at constant speed. The time window for successful trial is set to shorter than a person's typical visual reaction time (up to 250 ms) [18, 50]. The player must anticipate and plan the input so as to acquire the target successfully. A series of models for predicting user error rates in such anticipated input tasks has recently been published [24, 27–29, 35–37]. The moving-target selection model extended in this paper [28] is the latest of these.

In a task that requires anticipation for input, the moment the system responds may lag behind or jump ahead of the timing anticipated by the user. If the system response is *faster* than the user's anticipation, a non-intuitive conclusion follows: that latency must be *increased* for minimizing the discordance. This effect has recently been reported by several authors [24, 27–29]. In their studies, delaying the system response of the button-pressing reduced the user's error rate by 5–94%. The effect of latency is more complicated in an anticipatory task than in a reactive task, and, to the best of our knowledge, there remains no decent model to explain and predict it.

[2]The number of kills that a player made in average one minute.

Reducing Effects of Latency

Reducing the amount of end-to-end latency is the most direct way to decrease the effect of latency [44]. Researchers have been improving hardware and software to minimize the total latency, by such means as considering the best locations for game servers [3] and developing software that eliminates delays from synchrony [52]. The main goal with these has been to reduce latency to an imperceivable level [34].

Trajectory prediction, such as dead reckoning, is another way to reduce perceived latency by predicting the best possible current movement from the history of tracked movements. To reduce the perceived delays in an head-mounted display environment, researchers have implemented a "look-ahead" algorithm to predict the head-tracking data [47, 51]. Using prediction, Knibbe et al. improved high-speed motion-projector–camera systems by such means as projecting images onto moving objects [25]. Recently, some researchers have applied trajectory prediction in combination with additional sensors such as accelerometers or inertial measurement units (IMUs) [2, 26]. Their approach is effective with ballistic motion but cannot be directly applied to a moving-target selection task, which is performed based on a user's small movement with a single button.

In FPS games, wherein real-time operation is crucial to player performance, "favoring the shooter" is a widely used latency-compensation method [30]. This method makes decisions on events in line with the game world that the shooter sees when shooting [30]. Although keeping players from aiming ahead of the targets, this has "shot around a corner" inconsistency issues [7, 30]. To eliminate the oddity wherein players get shot even though they hid behind walls, some games, such as *Battlefield 4* and *Overwatch* [39], impose a 250 ms maximum for their lag-compensation [30].[3] Rolling back the game clock is deemed *unfair* and *not playable* from the point of view of players who have been shot [40].

Summary

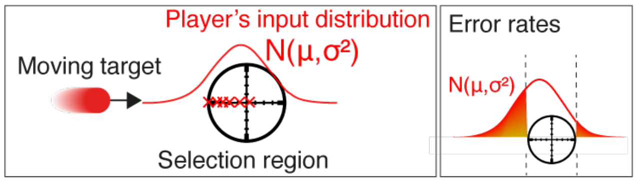
The effect of latency on usability has been considered primarily negative. However, for anticipated input tasks, which occur frequently in real-time games, latency exhibits a more complicated effect on user performance than is seen in reactive input tasks. Nonetheless, relatively simple approaches have been taken to compensate for latency in real-time games. We have complemented existing models for anticipated inputs, and the discussion below presents *geometric*

[3] Lagged players with a round-trip time (RTT) of 250 ms or more do not receive compensation via this method

compensation, a novel way to eliminate or control the effects of latency, accordingly.

3 MOVING-TARGET SELECTION TASK OVERVIEW

Moving-target selection is one of the most representative tasks in real-time games: players must activate the button input within a particular time window – for example, while a moving target is passing over a particular selection region. Since the time window is usually shorter than human reaction time, the player must anticipate the timing of the input before the target reaches that region. With multiple trials, player's input points can be approximated to be Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ in time. The error rate is then obtained by integrating the input point distribution outside the selection region as depicted in the figure below:



In the most recent model for moving-target selection [28], multiple factors influence μ and σ (see the Table below). The signs in parentheses in the table indicate whether σ or μ increases or decreases as the corresponding factor increases. The model proposed in this study adds the effect of end-to-end latency (L) to the table. The following section first introduces known factors in the existing model that affect μ and σ of the input point distribution. The section after that introduces new factors (marked in red) related to latency.

	Affecting σ	**Affecting μ**
Task factor	$t_c(-)$, $P(+)$, $L(+)$	$W_t(+)$, $L(+)$
Player factor	$c_\sigma(+)$, $\nu(-)$, $\delta(+)$	$c_\mu(+)$, $c_L(+)$

Factors Affecting the Standard Deviation (σ)

Task design has a huge impact on the standard deviation (σ) of the user-input distribution. The following two factors exist in this regard:

- **Period of input repetition** (P): If the inputs are repeated with a period of P, the player can anticipate the moment of the upcoming input. The longer the period of input repetition is, the larger the standard deviation (σ) of the input distribution for the player becomes. This phenomenon is connected with the scalar property of the internal clock [9, 14]. Recall that clapping once every five seconds is much harder than clapping once every second.
- **Cue-viewing time** ($\mathbf{t_c}$): This is the time from when the moving target starts to become visible until it reaches the selection region. The shorter t_c is, the more difficult it is for the user to anticipate timing information from the target's motion, and the standard deviation (σ) of the input distribution increases.

Also, player-side factors influence the σ:

- **Precision of the player's internal clock** ($\mathbf{c_\sigma}$): Even with the same amount of information, anticipation performance vary by individual. The parameter c_σ aggregates such user-side factors. The higher its value, the less precise the player's internal clock is and, thereby, the higher the σ of the input distribution.
- **Drift rate** (ν): The rate at which the player encodes timing information visually from a moving target varies too. A player with high ν produces an input distribution with a lower σ for the same cue-viewing time t_c.
- **Precision limits in motion decoding** (δ): No matter how long the cue-viewing time (t_c) given, the timing precision that a user can obtain from a moving target is limited. δ represents the limit. A player with a lower δ produces an input distribution with a lower σ for the same given t_c.

Factors Affecting the Mean (μ)

The input distribution's mean value, μ, is known to vary with the following factors:

- **Target-selectable duration** ($\mathbf{W_t}$): This is the duration of the time window where a moving target is on the selection region. It can be obtained by dividing the spatial size of the selection region by the speed of the target.
- **Implicit aim point** ($\mathbf{c_\mu}$): This is the user's implicit aim point within the selection region, which is known to a constant ratio of aim point to the size of the selection region [29], (μ=$c_\mu \cdot W_t$). That is, if c_μ is 0.5, the player always aims at the center of the given selection region.

Error Rates in Moving-Target Selection

Let's define $t = 0$ as the time at which the target contacts the selection region for the first time. The time window where the target stays in the selection region can be expressed as $[0, W_t]$. The error rate (E) is then obtained by integrating the input distribution of the player $\mathcal{N}(\mu, \sigma^2)$ over the time interval outside the selection region. According to the previous study [28], this results in the following equation:

$$E = 1 - \frac{1}{2}\left[erf\left(\frac{(1-c_\mu)}{c_\sigma\sqrt{2}} \cdot \frac{W_t}{D_t}\right) + erf\left(\frac{c_\mu}{c_\sigma\sqrt{2}} \cdot \frac{W_t}{D_t}\right)\right] \qquad (1)$$

D_t is used in the denominator in line with this equation:

$$D_t = P/\sqrt{1 + (P/(1/(e^{\nu t_c} - 1) + \delta))^2} \qquad (2)$$

erf (error function) is a special function that often appears in the integration process of the Gaussian distribution. Refer to the original paper [28] for the detailed derivation process.

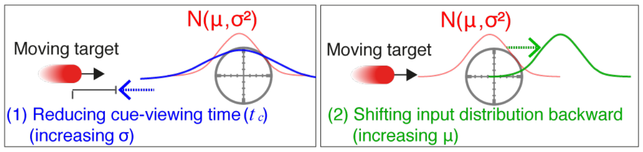

N(μ,σ²)
Moving target
(1) Reducing cue-viewing time (t_c)
(increasing σ)
N(μ,σ²)
Moving target
(2) Shifting input distribution backward
(increasing μ)

Figure 3: Effects of latency on moving-target selection task

4 THE NEW MODEL REGARDING THE EFFECT OF END-TO-END LATENCY

On top of the existing model above, we derived an advanced model that accounts for the effects of latency on player's input distribution. The new model assumes that latency has two effects on the moving-target selection (see Figure 3).

Firstly, a player experiences a decrease in cue-viewing time. With a certain amount of latency, the player must prepare and execute the input earlier compared to a no-latency condition. This preparation reduces the amount of time for the player to observe the target's movement. The assumption can be expressed as:

$$t_{eff} = MAX\,[(t_c - L), 0] \tag{3}$$

Here, t_c is the original cue-viewing time and t_{eff} is the newly calculated *effective* cue-viewing time under the influence of latency L. MAX function is applied because t_{eff} should never be negative even with a great latency.

Secondly, latency shift the input distribution backward in time. However, the extent of this may vary from player to player, so the assumption can be expressed as follows:

$$\mu_{eff} = \mu + c_L \cdot L \tag{4}$$

Here, μ is the average from the player input distribution when no latency is present and μ_{eff} is the mean of the distribution under latency L. c_L is a player-specific parameter that indicates how much the mean is affected by latency. Substituting the above formula into Equation 1 yields the new moving-target selection model that considers latency:

$$E = 1 - \frac{1}{2}\left[erf\!\left(\frac{(1-c_\mu)W_t - c_L L}{c_\sigma\sqrt{2}\,D_t}\right) + erf\!\left(\frac{c_\mu W_t + c_L L}{c_\sigma\sqrt{2}\,D_t}\right)\right] \tag{5}$$

The D_t with latency effect included is expressed thus:

$$D_t = P/\sqrt{1 + (P/(1/(e^{\nu(t_{eff}-L)} - 1) + \delta))^2} \tag{6}$$

Geometric Compensation of Latency Effect

We propose a geometric compensation method when varying latency exists in the system. It controls the error rate by slightly changing the geometric design of the game.

The full process has four steps (see Figure 4). **(1)** Obtain the **player-side factor**s of the model ($c_\sigma, c_\mu, \nu, \delta, c_L$) for the target game. This can be achieved by playing the game under various conditions, finding the case-specific error rates, and fitting them to the model (Equation 5). In order for fitting to be successful, the model must observe error rates for at least four to six task conditions. Assuming that 50 trials for each condition, about 300 button inputs are required. This can be done individually for each player or at a population-level for a particular game. This study focuses on showing the possibility of compensation at the population-level. **(2)** Identify the model's **task factors** (P, t_c, W_t, L) in the current game design. P, W_t, t_c are set by a designer. L of a system could be measured or estimated. **(3)** Set the game's difficulty in terms of desired players' error rate (target error rate). **(4)** Using Equation 5, derive new W_t and t_c that minimizes the difference between the current error rate and the target error rate. This can be completed with any nonlinear optimization toolbox. Those values alter the game's post-compensation geometric appearance. For example, W_t can be adjusted by changing the size of the selection region. Also, t_c can be adjusted by changing the size of the area that makes the target invisible (e.g., wall or curtain). Note that P and L are *temporal* factors and not altered.

5 STUDY 1: MODEL VALIDATION

In this experiment, the proposed moving-target selection model was verified via a custom-built device with precise control of latency. A point target moves at constant speed toward a one-dimensional selection region. Participants were asked to catch the target by pressing a button.

Method

Participants: Sixteen participants (3 female and 13 male) were recruited from a local university. Their average age was 24.9 years (SD=3.5). Each was paid $10 for 50 minutes of participation. Participants were informed and signed a consent form before the experiment.

Experimental Design: The experiment was a 2×2×5×2 within-subject design with four independent variables: *Period* of input repetition (P), *Target-Selectable Duration* (W_t), end-to-end *Latency* (L), and *Cue-Viewing Time* (t_c):

- Period (P): 1,250 and 1,800 ms

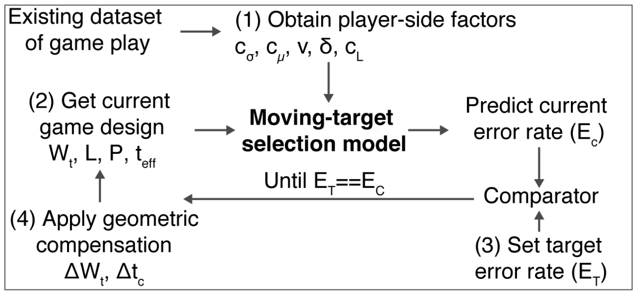

Existing dataset of game play
(1) Obtain player-side factors
c_σ, c_μ, ν, δ, c_L
(2) Get current game design
W_t, L, P, t_eff
Moving-target selection model
Predict current error rate (E_c)
Until E_T==E_C
Comparator
(4) Apply geometric compensation
ΔW_t, Δt_c
(3) Set target error rate (E_T)

Figure 4: Flow of geometric compensation process

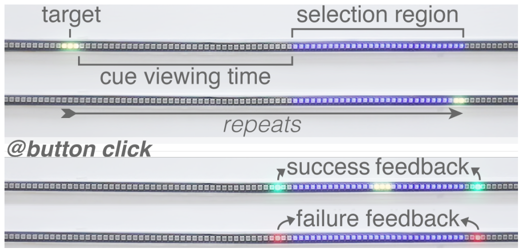


Figure 5: The task in Study 1, to select a moving target within the selection region by clicking a button, as the target moved from left to right.

- Target-Selectable Duration (W_t): 80 and 150 ms
- Latency (L): 0, 60, 80, 130, and 200 ms (+1.5 ms baseline)
- Cue-Viewing Time (t_c): 0 and 200 ms

Task: Participants were asked to complete the task of activating a button when a moving target point was in the one-dimensional selection region. The target moved in a straight line, on an LED strip attached horizontally to a wall. The speed of the target was fixed at 1.39 m/s (200 LED/s). If the button was activated while the target was in the selection region, the trial was considered successful; otherwise, a failure was recorded. With a given *Period*, the target repeatedly appeared from the left. *Cue-Viewing Time*, t_c, was the time for which the target was visible before reaching the selection region. The physical length of the selection region and cue-viewing region were calculated from the given W_t, t_c, and target speed (i.e., length of selection region = $W_t \times$ target speed). When the participant pressed the button, the activation event was delayed by the given *Latency*. Feedback for each button press came as blinking *green or red* lights along both sides of the selection region, for *success or failure*, respectively (see Figure 5).

Apparatus: We implemented a one-dimensional moving-target selection device with precisely controlled latency (check the supplement material for the detail). To minimize default latency, a commercial LED strip (Adafruit DotStar[4], 3 meters, 144 LED/m) was used as the display. A dedicated LED strip driver board (`Arduino Uno` using the Adafruit DotStar Library[5]) drove 300 LEDs on the strip at 333.3 Hz. The moving target was moderate yellow in color (#202000), and the color value of the selection region was dark blue (#000001). Our feedback colors were bright green (#002001) for success and bright red (#200000) for failure.

An experiment board (`Arduino 101`) sets experimental conditions for the LED strip driver board through UART serial communication. It also monitored the following events through digital interconnection wires: LED target appearing/disappearing, LED target entering/exiting the selection region, and button press/release. For user input, the left button of a dummy mouse (activated at 0.4 mm with 55 cN force) was directly connected to the experiment board. The board recorded timestamps of all events internally. The board transmitted the events (with timestamps, to 0.1 ms precision) to a PC (Mac mini Late 2014, 2.6 GHz Intel Core i5) for logging. The device exhibited 1.5 ms of mean latency.

To implement latency, the experiment driver board maintained a circular queue. Switch state (opened or closed) of the mouse button was enqueued every 0.5 ms with timestamp. If items in the queue exceed the *Latency* time, they were dequeued and triggered the press/release events. By this method, all sequences of the button events were precisely delayed. The baseline latency was not considered, so 1.5 ms should be added to the experimental latency values.

Setup and Procedure: Participants were asked to sit on a regular office chair 2.8 meters away from the LED strip. The LED strip was installed horizontally at subjects' eye level. The dummy mouse was on a desk in front of the chair. After completing the pre-experiment questionnaire, participants were given a practice session to familiarize themselves with the task. Participants were instructed to try not to skip any of the repeating targets. Forty task conditions (2×2×5×2) were randomly assigned to the participants. There were 40 trials for each condition, and a minimum of one-minute break was given between conditions. Accordingly, there were 1,600 trials per participant, taking 50 minutes. The apparatus logged the result of each trial and the timing of every events.

Results

In total, 25,600 trials were logged from 16 participants. The overall error rate measured for participants in the experiment was 45% (SD=23%). For statistical testing, we used repeated-measure ANOVA with an alpha level of 0.05. Greenhouse–Geisser correction was used for violation of sphericity. As forty task conditions were progressed, no increase in error rate due to fatigue was observed ($F(1,15)=0.838$, $p=0.374$).

Normality of the Response Distribution: Before analyzing the results, we consider whether the timing distribution obtained in this experiment shows a Gaussian distribution as expected. We tested the normality of each task condition block. A one-sample Kolmogorov–Smirnov test was performed for 640 blocks (40 conditions × 16 participants) in total ($\alpha = .05$).

Of the 640 distributions obtained (one per condition per user), 70.3% were statistically normal distributions. This is similar to the level observed in typical timing tasks [29]. For the participants who met the normality condition at the lowest rate, 60% of the conditions were found to be statistically

[4]https://learn.adafruit.com/adafruit-dotstar-leds/overview
[5]https://github.com/adafruit/Adafruit_DotStar

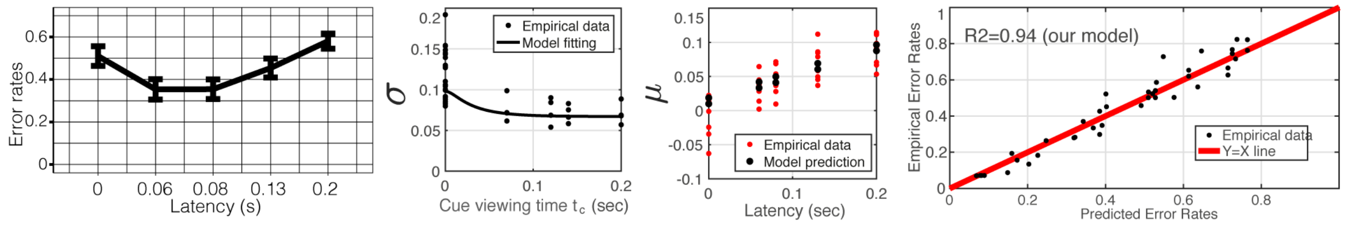


Figure 6: Results of Study 1

normal. Since this is not a particularly problematic level, all subsequent analyses included data from all participants.

Effect of 'P, W_t, and t_c' on Error Rates: Their effect on the error rate of the moving-target selection task have been reported in detail in previous studies [28, 29]. Below, we replicate each effect once again and report on it briefly.

Firstly, all factors had statistically significant effects on the error rate: P ($F(1,15)$=106.4, $p < 0.001$), t_c ($F(1,15)$=485.1, $p < 0.001$), and W_t ($F(1,15)$=750.1, $p < 0.001$). We replicated the tendency for error rates to rise as *Period* (P) increases, *Cue-Viewing Time* (t_c) decreases, and *Target-Selectable Duration* (W_t) decreases.

Effect of Latency on Error Rates: End-to-end latency too had a significant impact on the error rate ($F(2.205,33.079)$=53.975, $p < 0.001$). Contrary to the conventional belief that error rate will increase when latency rises, a slight latency seems to benefit performance of tasks involving anticipation by the user, such as moving-target selection (see Figure 6). The condition with 60–80 ms latency added showed an error rate that was 15.7% lower (mean=35.3%, SD=27.2%) than that in the zero-latency condition (mean=51%, SD=26.4%).

Model Fitting

Testing Assumptions about Effect of Latency on σ and μ: We validated Section 4's assumptions related to latency for input-distribution σ and μ (expressed as equations 3, 6, and 7) by fitting the equations to the data obtained. All the curve fittings were carried out with the `patternsearch` function provided by `MATLAB Global Optimization Toolbox`. Fitting was performed at the population-level by averaging all participant data for each condition.

As a result, the predicted σ value and the actually measured σ are fitted with a high correlation (R^2=0.77; see Figure 6). Also, the phenomenon of input-distribution mean μ shifting with latency is explained by the model with a moderate coefficient of determination (R^2=0.70; see Figure 6).

Error Rates: The proposed model is ultimately aimed at predicting the error rate displayed by users in the given task condition. This is achieved by plugging the independent variables and the observed error rate into Equation 5.

Hence, the model was fitted with the observed error rate and showing a high coefficient of determination (R^2=0.94; see Figure 6). If the effect of latency is not considered, the earlier model for moving-target selection [28] provides a low fit (R^2=0.65). The five free parameters obtained from the fitting are summarized in Table 1. For reference, individual-level fitting was attempted for each participant, and the results are as follows: c_μ=0.056 (SD=0.099), c_σ=0.091 (SD=0.017), ν=19.54 (SD=21.77), δ=0.243 (SD=0.202), c_L=0.53 (SD=0.16), R^2=0.81 (SD=0.051).

Discussion

In the moving-target selection task with latency added, the proposed model faithfully predicted the empirical error rate of the user. In particular, in the study we observed a decline in participants' error rate with latency increases up to a certain level. To achieve the lowest error rate in the timing task, players must aim at the center of the selection region as the Gaussian distribution is symmetric about the mean. However, users are generally known to aim at the front of the selection region (note that c_μ values in Table 1 are lower than 0.5). This is called the negative mean asynchrony (NMA) phenomenon [42]. The increase in latency allows the user's aim point to move closer to the center of the selection region. So, while not intuitive, an increase in latency can lead to a reduction in the error rate. This illustration that the system's response can sometimes be too fast in a task involving participant anticipation of input timing replicates and theoretically extends the results of some studies previously reported upon [24, 27–29].

Table 1: Summary of empirical parameters

Studies	c_μ	c_σ	ν	δ	c_L	R^2
Study 1 with our model	0.061	0.089	10.59	0.18	0.50	0.94
Study 1 with CHI'18 model [28]	0.35	0.095	49.48	0.91	N/A	0.65
2016 Flappy Bird data [29] our model assuming 30 ms latency	0.089	0.071	66.53	0.23	0.14	0.98
Study 2 with our model	0.087	0.027	5.97	0.49	0.0091	0.94

The model's assumptions about the effects of latency on the input distribution's μ and σ values were verified. Thus, we can report that the model succeeded in predicting participants' empirical error rates with high accuracy (R^2=0.94).

We now turn to the empirical parameters obtained through fitting. The previous model for moving-target selection [28] failed to capture the tendency of latency to push back the mean of the input distribution, μ. The effect was reflected instead in the measurement of the user's implicit aim point (c_μ). For the same dataset, the prior model produces a large c_μ value, about 0.35 (see Table 1), which results from misinterpreting the latency-induced shift in input distribution to be a shift in the user's intended aim point.

We attempted to further analyze whether the phenomenon of shifting the input distribution backward is a passive effect of latency vs. intended by the user. If we assume that c_μ is directly affected by latency, the μ of the input distribution can be expressed thus:

$$\mu = c_\mu \cdot W_t = (c_{\mu 0} + c_L \cdot L) \cdot W_t \tag{7}$$

Fitting this equation to the μ of the empirically measured input distribution yields a worse fit (R^2=0.47) than our model's (R^2=0.70). This supports the view that the input-distribution shift represents not user intention but a passive effect of latency.

However, there are hypotheses that cannot be answered by this analysis alone. For example, to achieve a lower error rate, the user may tolerate the input being delayed without actively correcting the aim point. In contrast, under experimental conditions wherein the error rate can be critically increased via latency (i.e., by latency above 200 ms), the user may actively modify the implicit aim point. In fact, it has been observed that as the latency value approaches 200 ms, the change in μ value saturates (see the third graph in Figure 6). A definitive answer is beyond the scope of this study and requires further investigation.

6 STUDY 2: GEOMETRIC COMPENSATION OF LATENCY

For the famous moving-target selection game Flappy Bird, Study 2 demonstrates the geometric compensation of the latency effect.

Moving-Target Selection in Flappy Bird

In Flappy Bird, the player must control the flight of a small bird and pass it through pillars that each have a narrow gap. The player must activate a button occasionally to control the bird's flight: when this is pressed, the bird starts rising at a constant speed, but if there is no further input it then falls to the ground under gravity. The game ends when the bird hits the ground or a pillar.

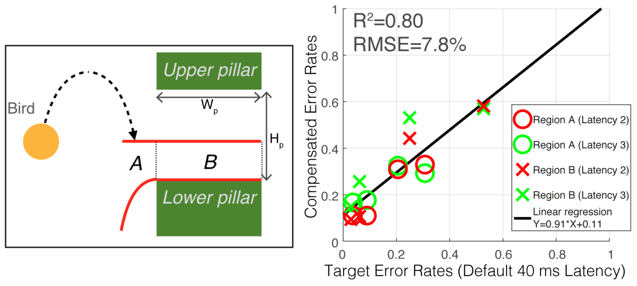


Figure 7: Two selection regions in the Flappy Bird game (left), the result of the geometric compensation performed by adjusting the pillar spacing (right)

Keeping the bird from falling to the ground is easy, so this task was not considered in our study. However, the player must perform a moving-target selection task to pass through a pillar without bumping into it. At this time, the player encounters two selection regions in the game scene, of different shapes (see Figure 7).

Geometric Compensation in Flappy Bird

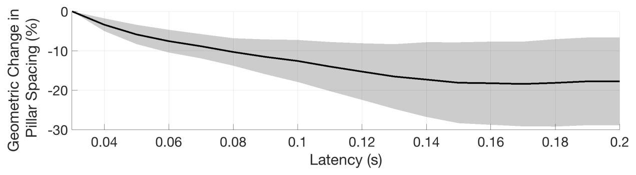
The difficulty level in Flappy Bird is determined by the speed of the bird, the vertical spacing between pillar sections, and the magnitude of the gravity. Of these variables, the one that determines the look of the game geometry is the spacing between the pillars. Other variables must be kept constant to maintain the temporal dynamics of the game.

When the vertical spacing for a pillar is changed, target-selectable duration W_t, which determines the user's error rate, is changed. Input period P and cue-viewing time t_c do not change when the bird's speed and gravity are constant. In Study 2, we controlled the player's error rate by adjusting the vertical spacing of pillar segments to change W_t.

How vertical gaps determine the W_t value is calculated from an equation obtained in previous work [29] (not repeated here). The change in the player's error rate due to the change in W_t can be predicted via Equation 5. This enabled geometric compensation to be applied for Flappy Bird in line with Figure 4. Changing the pillar gaps alters the W_t values in selection regions A and B both (see Figure 7).

Obtaining Empirical Parameters

The 10 participants in the aforementioned study [29] played the Flappy Bird game under various W_t, P, and t_c conditions. Using their dataset, we fitted our model to obtain empirical parameters. Since we did not know the end-to-end latency of the apparatus used in the earlier experiments (`MacBook Pro` late 2012, 2.2 GHz Intel Core i7, Intel Iris Pro 1536 MB graphics card). Therefore, it was assumed that the typical latency of a MacBook Pro keyboard, 30 ms, was present in those experiments. Our model showed a high fit with their data (R^2=0.98). The resulting empirical parameters were used for geometric compensation in Study 2 (see Table 1). When the latency changes from 30 ms to 200 ms, the spacing between the pillars varies by compensation from -34.85% to 3.57% as shown in the figure below (the shading indicates the 95% confidence interval):



Method

Participants: Twelve participants, four of them female and eight male, were recruited from a local university. Their average age was 23.61 (SD=4.4), and each was rewarded with $10 for completing the experiment. Participants were informed and signed a consent form before the experiment.

Task: Participants were asked to play Flappy Bird implemented in Java (see Figure 8). They made the bird jump by pressing the spacebar. We asked the participants to do their best to avoid colliding with the pillars and make the bird fly as far as they could. The horizontal velocity of the bird was 3 pixels/frame, and the speed at which the bird rose upon pressing of the spacebar was 5 pixels/frame. The temporal dynamics of the final application were similar to those of the original Flappy Bird.

Apparatus: The game was implemented on a desktop computer (`Intel Core(TM) i7-7700 CPU @3.60 GHz, Windows 10`) and was constantly driven at a frame rate of 60Hz. A 24-inch monitor (`ASUS ROG Swift PG248Q 180 Hz`) and a gaming keyboard (`ASUS ROG Claymore`) were used. We measured the end-to-end latency of the system with a high-speed camera (960 fps, SONY DSC–RX100M5). The resulting latency was approximately 40 ms between when the keyboard button was pressed and feedback appeared on the screen.

Experimental Design: The experiment had a 4×3 within-subject design with two independent variables: *Game Condition*, and *Latency*. The levels were the following:

- Game Condition: *Condition 1* (Gravity g_1–0.11 pixels^2/frame, Pillar Width W_{p1}–272 pixels, Pillar Spacing H_{p1}–190 pixels), *Condition 2* (g_2–0.22 pixels^2/frame, W_{p2}–136 pixels, H_{p2} 132– pixels), *Condition 3* (g_3–0.22 pixels^2/frame, W_{p3}–136 pixels, H_{p3}–94 pixels), *Condition 4* (g_4–0.22 pixels^2/frame, W_{p4}–136 pixels, H_{p4}–84 pixels)
- Latency: *Latency 1* (40 ms, default latency), *Latency 1* (80 ms, 40 ms added), *Latency 3* (120 ms, 80 ms added)

The additional latency was implemented through software and was applied from the moment of the `keyPressed` event. For each *Latency–Game Condition* pair except with default

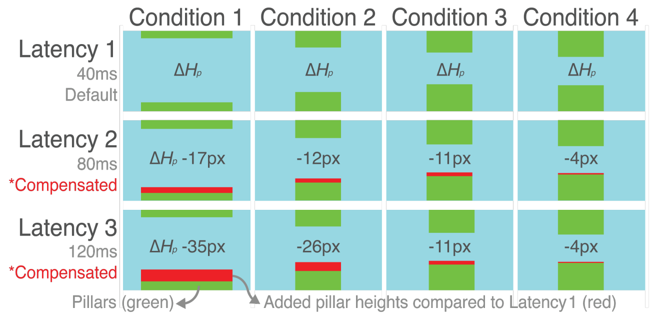


Figure 8: On successful compensation, error rates of Latency 1, 2 and 3 should be similar for each Condition.

latency, the following geometric compensation was applied to H_p: $\Delta H^{80}_{p1} = -17$ pixels, $\Delta H^{80}_{p2} = -12$ pixels, $\Delta H^{80}_{p3} = -11$ pixels, $\Delta H^{80}_{p4} = -4$ pixels, $\Delta H^{120}_{p1} = -35$ pixels, $\Delta H^{120}_{p2} = -26$ pixels, $\Delta H^{120}_{p3} = -11$ pixels, $\Delta H^{120}_{p4} = -4$ pixels. Since the latency used was expected to lower the participants' error rates, the pillar gaps had to be reduced overall after the geometric compensation was performed. After this compensation, the error rate for each *Game Condition* would be expected to remain the same across *Latency* conditions.

Setup and Procedure: Participants were instructed to sit on a gaming chair in front of the PC. During an experimenter's brief explanation of the experiment, it was verified that each subject understood the tasks. None of the participants were familiar with the original Flappy Bird game. The order of the three *Latency* conditions was counterbalanced via a Latin square design. In each *Latency* condition, the order of *Game Condition* assignments was random. To keep trial counts similar irrespective of *Game Condition*, participants played for six minutes with a gravity of 0.11 pixels^2/frame and three minutes with a gravity of 0.22 pixels^2/frame. All received $10 as compensation for participating.

Results

Let E_1 be the error rate in the default latency condition (*Latency 1*), E_2 be the error rate in the presence of 40 ms added latency (*Latency 2*), and E_3 be the error rate when additional latency of 80 ms is present (*Latency 3*). With the geometric compensation, the following relationship between these three error rates should be observed: $E_0 \approx E_{40}$ and $E_0 \approx E_{80}$.

The result is shown on the right in Figure 7. The x-axis of the graph represents the error rate measured under the *Latency 1* condition for each *Game Condition* setting. On the y-axis is the error rate obtained in the measurements when participants were exposed to the latency-compensation conditions (*Latency 2* and *Latency 3*). The graph illustrates that geometric compensation allows participants to maintain error rates similar to those with the default latency and geometry conditions ($R^2 = 0.80$ and root mean square error (RMSE)=7.8%).

Discussion

By means of empirical parameters obtained via Flappy Bird data published in 2016 [29], geometric compensation for additional latency was attempted. This was successful in terms of linearity (R^2=0.80) and RMSE (7.8%). Note that the compensation was performed at the population-level and did not take into account the effect of each player's difference. We discuss other factors that affected the compensation accuracy in Study 2.

Simultaneous Compensation of Multiple Selection Regions: As mentioned earlier in Figure 7, the game Flappy Bird features two types of selection region, and the change in intra-pillar spacing affects both shapes simultaneously. That is, if compensation for the error rate is applied for one kind of selection area, the error rate for the other may not reach the level targeted. In Study 2, we computed post-compensation pillar gaps for each of the two region types and averaged these to obtain a single compensation-inclusive spacing. This compromises the accuracy of the compensation. In fact, as Figure 7 shows, the post-compensation error rate for selection region A is slightly lower than the post-compensation error rate for selection region B.

If the speed of the bird too were adjusted, it would have been possible to compensate for the error rate from both selection regions simultaneously. However, this could affect the game's temporal dynamics; game designers should apply it only selectively.

Inaccurate Estimation of Empirical Parameters: For performing geometric compensation, the empirical parameters of the moving-target selection model '$c_\mu, c_\sigma, \nu, \delta, and\ c_L$' must be known in advance. In our experiment, these parameters' values were obtained by fitting the model to the Flappy Bird dataset from the earlier paper [29]. Again, since the authors did not report the end-to-end latency of the system from which the data were obtained, we made a rough estimate for our paper that puts the latency of the system in the 2016 Flappy Bird experiment at about 30 ms. If errors in this assumption led to the empirical parameters not being obtained correctly, the accuracy of the geometric compensation performed in Study 2 may have suffered.

For confirmation, we fitted the model with only the data obtained in Study 2 and acquired empirical parameters again. Since the latency in the newer user study is clearly known (40 ms, 80 ms, 120 ms), the empirical parameters obtained from the Study 2 dataset can be regarded as ground truth. Thus, we can conclude that our model successfully explained the error rate of the participants in Study 2 ($R^2 = 0.94$).

The resulting parameters differed from the set used to design the geometrical compensation in Study 2 (see Table 1). There are two possible causes for this difference: (1) the end-to-end latency in the 2016 experiment may *not* have been 30 ms, and (2) subjects' demographic differences may be relevant. However, no significant demographic differences were found between the participants in the 2016 experiment [29] and those in Study 2, so we conjecture that error exists in our assumption that the system latency in the 2016 experiment was 30 ms.

Inverse Modeling of Latency in 2016 Paper: Assuming that the empirical parameters obtained from the Study 2 dataset are ground truth, we can estimate the end-to-end latency in the 2016 Flappy Bird experiment [29] through *inverse modeling* [23]. The end-to-end latency in the earlier study was regarded as an unknown variable and fitted to our model via ground-truth empirical parameters. As a result, the model described the data well ($R^2 = 0.96$), and the 2016 system's latency was found to be *122* ms, a much larger value than we expected. This may be the reason for our geometrical compensation not being perfect.

7 LIMITATION AND CONCLUSION

Our research yielded a novel model and method that can compensate for the effects of latency by *geometrically transforming* the shape of the game scene. The idea could be further generalizable to other games. For example, followings are applicable geometric compensation methods on existing popular games: increase/decrease a route width in Dancing Line, enlarge/reduce a size of enemy in a FPS game, and adjust a ball size in Rolling Sky. The method does not interfere with the time progress of the game, so it can enhance the player's immersion unlike prior time rewinding methods. That said, our work was confined to games based on relatively simple tasks (i.e., moving-target selection) and did not test how geometric compensation actually benefits the player. Also, this study did not address how the geometric compensation of latency affects the quality of experience (QoE) of players. Nonetheless, the model proposed in this paper represents a low-level mechanism for the effects of latency and was derived on that basis, so follow-up studies should be able to easily extend the model to more complex tasks. Finally, our non-intuitive finding that the error rate may decrease with some latency increases is an important take-away, one with implications for future game design.

8 ACKNOWLEDGEMENTS

This research was supported by the National Research Foundation of Korea (2017R1C1B2002101, 2018R1A5A7025409). SK's research was funded by the Aalto University Seed Funding Grant GamerLab.

REFERENCES

[1] Glen Anderson, Rina Doherty, and Subhashini Ganapathy. 2011. User Perception of Touch Screen Latency. In *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, Aaron Marcus (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 195–202.

[2] Axel Antoine, Sylvain Malacria, and Géry Casiez. 2018. Using High Frequency Accelerometer and Mouse to Compensate for End-to-end Latency in Indirect Interaction. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 609, 11 pages. https://doi.org/10.1145/3173574.3174183

[3] G. Armitage. 2003. An experimental estimation of latency sensitivity in multiplayer Quake 3. In *The 11th IEEE International Conference on Networks, 2003. ICON2003.* 137–141. https://doi.org/10.1109/ICON.2003.1266180

[4] Raymond E. Barber and Henry C. Lucas, Jr. 1983. System Response Time Operator Productivity, and Job Satisfaction. *Commun. ACM* 26, 11 (Nov. 1983), 972–986. https://doi.org/10.1145/182.358464

[5] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. 2004. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003®. (2004), 144–151. https://doi.org/10.1145/1016540.1016556

[6] François Bérard and Renaud Blanch. 2013. Two Touch System Latency Estimators: High Accuracy and Low Overhead. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces (ITS '13)*. ACM, New York, NY, USA, 241–250. https://doi.org/10.1145/2512349.2512796

[7] Yahn W Bernier. 2001. Latency compensating methods in client/server in-game protocol design and optimization. In *Game Developers Conference*, Vol. 98033.

[8] Florian Bockes, Raphael Wimmer, and Andreas Schmid. 2018. LagBox – Measuring the Latency of USB-Connected Input Devices. , Article LBW115 (2018), 6 pages. https://doi.org/10.1145/3170427.3188632

[9] Catalin V Buhusi and Warren H Meck. 2005. What makes us tick? Functional and neural mechanisms of interval timing. *Nature Reviews Neuroscience* 6, 10 (2005), 755.

[10] Géry Casiez, Thomas Pietrzak, Damien Marchal, Sébastien Poulmane, Matthieu Falce, and Nicolas Roussel. 2017. Characterizing Latency in Touch and Button-Equipped Interactive Systems. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*. ACM, New York, NY, USA, 29–39. https://doi.org/10.1145/3126594.3126606

[11] Irina Ceaparu, Jonathan Lazar, Katie Bessiere, John Robinson, and Ben Shneiderman. 2004. Determining Causes and Severity of End-User Frustration. *International Journal of Human-Computer Interaction* 17, 3 (2004), 333–356. https://doi.org/10.1207/s15327590ijhc1703_3

[12] Mark Claypool, Ragnhild Eg, and Kjetil Raaen. 2017. Modeling user performance for moving target selection with a delayed mouse. In *International Conference on Multimedia Modeling*. Springer, 226–237.

[13] Jonathan Deber, Bruno Araujo, Ricardo Jota, Clifton Forlines, Darren Leigh, Steven Sanders, and Daniel Wigdor. 2016. Hammer Time!: A Low-Cost, High Precision, High Accuracy Tool to Measure the Latency of Touchscreen Devices. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2857–2868. https://doi.org/10.1145/2858036.2858394

[14] John Gibbon, Russell M Church, and Warren H Meck. 1984. Scalar timing in memory. *Annals of the New York Academy of sciences* 423, 1 (1984), 52–77.

[15] Niels Henze and Benjamin Poppinga. 2012. Measuring latency of touch and tactile feedback in touchscreen interaction using a mobile game. In *Proceedings of the 3rd International Workshop on Research in the Large. New York: ACM. S.* 23–26.

[16] Zenja Ivkovic, Ian Stavness, Carl Gutwin, and Steven Sutcliffe. 2015. Quantifying and Mitigating the Negative Effects of Local Latencies on Aiming in 3D Shooter Games. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 135–144. https://doi.org/10.1145/2702123.2702432

[17] Aditya Jain, Ramta Bansal, Avnish Kumar, and KD Singh. 2015. A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students. *International Journal of Applied and Basic Medical Research* 5, 2 (2015), 124.

[18] Arthur R. Jensen and Ella Munro. 1979. Reaction time, movement time, and intelligence. *Intelligence* 3, 2 (1979), 121 – 126. https://doi.org/10.1016/0160-2896(79)90010-2

[19] Ricardo Jota, Albert Ng, Paul Dietz, and Daniel Wigdor. 2013. How Fast is Fast Enough?: A Study of the Effects of Latency in Direct-touch Pointing Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2291–2300. https://doi.org/10.1145/2470654.2481317

[20] T. Kaaresoja, E. Anttila, and E. Hoggan. 2011. The effect of tactile feedback latency in touchscreen interaction. In *2011 IEEE World Haptics Conference*. 65–70. https://doi.org/10.1109/WHC.2011.5945463

[21] Topi Kaaresoja and Stephen Brewster. 2010. Feedback is... Late: Measuring Multimodal Delays in Mobile Device Touchscreen Interaction. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction (ICMI-MLMI '10)*. ACM, New York, NY, USA, Article 2, 8 pages. https://doi.org/10.1145/1891903.1891907

[22] Teemu Kämäräinen, Matti Siekkinen, Antti Ylä-Jääski, Wenxiao Zhang, and Pan Hui. 2017. A Measurement Study on Achieving Imperceptible Latency in Mobile Cloud Gaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17)*. ACM, New York, NY, USA, 88–99. https://doi.org/10.1145/3083187.3083191

[23] Antti Kangasrääsiö, Kumaripaba Athukorala, Andrew Howes, Jukka Corander, Samuel Kaski, and Antti Oulasvirta. 2017. Inferring cognitive models from data using approximate Bayesian computation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 1295–1306.

[24] Sunjun Kim, Byungjoo Lee, and Antti Oulasvirta. 2018. Impact Activation Improves Rapid Button Pressing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 571, 8 pages. https://doi.org/10.1145/3173574.3174145

[25] Jarrod Knibbe, Hrvoje Benko, and Andrew D. Wilson. 2015. Juggling the Effects of Latency: Software Approaches to Minimizing Latency in Dynamic Projector-Camera Systems. (2015), 93–94. https://doi.org/10.1145/2815585.2815735

[26] Huy Viet Le, Valentin Schwind, Philipp Göttlich, and Niels Henze. 2017. PredicTouch: A System to Reduce Touchscreen Latency Using Neural Networks and Inertial Measurement Units. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS '17)*. ACM, New York, NY, USA, 230–239. https://doi.org/10.1145/3132272.3134138

[27] Byungjoo Lee, Qiao Deng, Eve Hoggan, and Antti Oulasvirta. 2017. Boxer: A Multimodal Collision Technique for Virtual Objects. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction (ICMI 2017)*. ACM, New York, NY, USA, 252–260. https://doi.org/10.1145/3136755.3136761

[28] Byungjoo Lee, Sunjun Kim, Antti Oulasvirta, Jong-In Lee, and Eunji Park. 2018. Moving Target Selection: A Cue Integration Model. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 230, 12 pages. https://doi.org/10.1145/3173574.3173804

[29] Byungjoo Lee and Antti Oulasvirta. 2016. Modelling Error Rates in Temporal Pointing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1857–1868. https://doi.org/10.1145/2858036.2858143

[30] Steven W. K. Lee and Rocky K. C. Chang. 2017. On "shot around a corner" in first-person shooter games. In *2017 15th Annual Workshop on Network and Systems Support for Games (NetGames)*. 1–6. https://doi.org/10.1109/NetGames.2017.7991545

[31] Jiandong Liang, Chris Shaw, and Mark Green. 1991. On Temporal-spatial Realism in the Virtual Reality Environment. In *Proceedings of the 4th Annual ACM Symposium on User Interface Software and Technology (UIST '91)*. ACM, New York, NY, USA, 19–25. https://doi.org/10.1145/120782.120784

[32] I. Scott MacKenzie and Colin Ware. 1993. Lag As a Determinant of Human Performance in Interactive Systems. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93)*. ACM, New York, NY, USA, 488–493. https://doi.org/10.1145/169059.169431

[33] Gale L. Martin and Kenneth G. Corl. 1986. System response time effects on user productivity. *Behaviour & Information Technology* 5, 1 (1986), 3–13. https://doi.org/10.1080/01449298608914494

[34] Albert Ng, Julian Lepinski, Daniel Wigdor, Steven Sanders, and Paul Dietz. 2012. Designing for Low-latency Direct-touch Input. (2012), 453–464. https://doi.org/10.1145/2380116.2380174

[35] Antti Oulasvirta, Sunjun Kim, and Byungjoo Lee. 2018. Neuromechanics of a Button Press. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 508, 13 pages. https://doi.org/10.1145/3173574.3174082

[36] Eunji Park, Hyunju Kim, and Byungjoo Lee. 2018. Button++: Designing Risk-aware Smart Buttons. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, LBW116.

[37] Eunji Park and Byungjoo Lee. 2018. Predicting Error Rates in Pointing Regardless of Target Motion. *arXiv preprint arXiv:1806.02973* (2018).

[38] Andriy Pavlovych and Wolfgang Stuerzlinger. 2009. The Tradeoff Between Spatial Jitter and Latency in Pointing Tasks. In *Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '09)*. ACM, New York, NY, USA, 187–196. https://doi.org/10.1145/1570433.1570469

[39] PlayOverwatch. 2016. Developer Update | Let's Talk Netcode | Overwatch. https://www.youtube.com/watch?v=vTH2ZPgYujQ

[40] Peter Quax, Patrick Monsieurs, Wim Lamotte, Danny De Vleeschauwer, and Natalie Degrande. 2004. Objective and Subjective Evaluation of the Influence of Small Amounts of Delay and Jitter on a Recent First Person Shooter Game. In *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games (NetGames '04)*. ACM, New York, NY, USA, 152–156. https://doi.org/10.1145/1016540.1016557

[41] K. Raaen, R. Eg, and C. Griwodz. 2014. Can gamers detect cloud delay?. In *2014 13th Annual Workshop on Network and Systems Support for Games*. 1–3. https://doi.org/10.1109/NetGames.2014.7008962

[42] Bruno H Repp. 2005. Sensorimotor synchronization: a review of the tapping literature. *Psychonomic bulletin & review* 12, 6 (2005), 969–992. https://doi.org/10.3758/bf03206433

[43] Walter Ritter, Guido Kempter, and Tobias Werner. 2015. User-Acceptance of Latency in Touch Interactions. In *Universal Access in Human-Computer Interaction. Access to Interaction*, Margherita Antona and Constantine Stephanidis (Eds.). Springer International Publishing, Cham, 139–147. https://doi.org/10.1007/978-3-319-20681-3_13

[44] Steven C Seow. 2008. *Designing and engineering time: The psychology of time perception in software*. Addison-Wesley Professional.

[45] Jose Shelton and Gideon Praveen Kumar. 2010. Comparison between auditory and visual simple reaction times. *Neuroscience & Medicine* 1, 1 (2010), 30–32.

[46] Ben Shneiderman. 1984. Response Time and Display Rate in Human Performance with Computers. *ACM Comput. Surv.* 16, 3 (Sept. 1984), 265–285. https://doi.org/10.1145/2514.2517

[47] Richard H.Y. So and Michael J Griffin. 1992. Compensating lags in head-coupled displays using head position prediction and image deflection. *Journal of Aircraft* 29, 6 (1992), 1064–1068.

[48] Steven L. Teal and Alexander I. Rudnicky. 1992. A Performance Model of System Delay and User Strategy Selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*. ACM, New York, NY, USA, 295–305. https://doi.org/10.1145/142750.142818

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

[50] Simon Thorpe, Denis Fize, and Catherine Marlot. 1996. Speed of processing in the human visual system. *nature* 381, 6582 (1996), 520. https://doi.org/10.1038/381520a0

[51] Jiann-Rong Wu and Ming Ouhyoung. 2000. On latency compensation and its effects on head-motion trajectories in virtual environments. *The Visual Computer* 16, 2 (01 Mar 2000), 79–90. https://doi.org/10.1007/s003710050198

[52] Min Hong Yun, Songtao He, and Lin Zhong. 2017. Reducing Latency by Eliminating Synchrony. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 331–340. https://doi.org/10.1145/3038912.3052557