The Impact of Motion and Delay on Selecting Game Targets with a Mouse

MARK CLAYPOOL, Worcester Polytechnic Institute, USA
ANDY COCKBURN, University of Canterbury, New Zealand
CARL GUTWIN, University of Saskatchewan, Canada

All real-time computer games, particularly networked computer games, have a delay from when a player starts an action (e.g., clicking the mouse) until the game renders the result (e.g., firing a projectile). This delay can degrade both player performance (e.g., reduced game score) and quality of experience (e.g., the game is less fun). While previous work has studied the effects of delay on commercial games and individual game actions, a more detailed understanding is needed of the effects of delay on moving target selection with realistic target motion, a common scenario in many games. This paper presents an in-depth study of the effects of delay on the fundamental game action of selecting a moving target with a mouse while varying two parameters for the target motion – turn frequency and turn angle. We design and implement a custom game where players select moving targets using a mouse, while the game controls both the target motion and input delay. Analysis of data gathered in a 56-person user study shows both target selection time and accuracy degrade with delay. However, both selection time and accuracy increase with the frequency and angle of changes in the target's movement, because turning slows targets down even while making them less predictable. We set these results in the context of other studies of delay and target selection by comparing our findings to those in seven other previously published papers that investigated the effects of delay on target selection.

CCS Concepts: • **Applied computing** → **Computer games**; • **Human-centered computing** → *User studies*;

Additional Key Words and Phrases: Lag, user study, target selection

ACM Reference format:
Mark Claypool, Andy Cockburn, and Carl Gutwin. 2020. The Impact of Motion and Delay on Selecting Game Targets with a Mouse. *ACM Trans. Multimedia Comput. Commun. Appl.* 16, 2s, Article 73 (June 2020), 24 pages.
https://doi.org/10.1145/3390464

1 INTRODUCTION

The computer games market is expected to be worth over 90 billion U.S. dollars by 2020, up from nearly 79 billion in 2017 (Pricewaterhouse Coopers, BestTheNews, 2016). There are more than 2.5 billion computer gamers worldwide (The European Mobile Game Market, 2016) who are

Authors' addresses: M. Claypool, Worcester Polytechnic Institute, Worcester, Massachusetts, USA; email: claypool@cs.wpi.edu; A. Cockburn, University of Canterbury, Christchurch, New Zealand; email: andy@cosc.canterbury.ac.nz; C. Gutwin, University of Saskatchewan, Saskatoon, Saskatchewan, Canada; email: gutwin@usask.ca.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Association for Computing Machinery.

1551-6857/2020/06-ART73 $15.00
https://doi.org/10.1145/3390464

increasingly playing games online. Online games alone are projected to be worth about 30 billion U.S. dollars (Capcom, International Development Group, 2017) and online PC games are expected to take 47% of the global PC and console gaming revenues in 2019 (DigiWorld, 2016). Of particular interest for studies of delay are cloud-based games, where a remote server runs the game and sends game images to the client, which records and sends user input back up to the server. This means that all input from the local device is subject to any delay created by the network conditions – which is different from typical online games where input is local and latency only affects the synchronization of the game state. Cloud-based games emerged in the early 2010s with services such as *OnLive* and Sony's *PlayStation Now*, but have seen a particular resurgence in 2020 with new services such as Google's *Stadia* and Microsoft's *xCloud*.[1]

Online games must deal with delay since player game input travels from the player's local computer and game world to game servers and other players, encountering processing and network delays on end-host computers and intermediate network devices. For traditional online games, network delays can manifest in inconsistent state between the clients and servers. For cloud-based games, network delays add at least a round-trip time from the client to the server in response to the player action [8]. Even non-networked games face local delays from the player action (e.g., moving the mouse) to rendering the effects in-game (e.g., cursor changing position) of up to 240 milliseconds [20]. Real-time games require players to execute many time-sensitive actions that degrade when delayed, and even delays less than a twentieth of a second can hamper the interplay between players' actions and intended results. For example, delay when aiming a virtual weapon in a shooting game can make it difficult for a player to hit a moving target, decreasing the player's score and degrading the quality of experience.

There are established methods to compensate for network delays in traditional online games [5, 20], such as system-level treatments (e.g., network packet priorities), delay compensation algorithms (e.g., dead reckoning, sticky targets, aim dragging) and even game designs that can mitigate perceived delay (e.g., deferred avatar response, geometric scaling). However, most of these techniques are not effective for local delays nor can all of them be used for cloud-based games. A detailed understanding of how delay affects player actions in games is needed in order to: (1) provide a better understanding for game developers who need to design levels and choose target motion parameters to account for delay, (2) provide insights that will aid in developing and deploying delay-mitigation techniques for local systems and cloud-based games, and (3) construct or improve models of the effects of delay on player actions useful for game designers.

Previous work has studied the effects of delay on commercial games, often clustering work based on genre – e.g., First Person Shooter [2, 4, 33], Massively Multiplayer Online [7, 16], Real-Time Strategy [9], and Sports [29, 30]. While helpful for providing insights into the effects of delay for traditional network games, such studies generally cannot tease out specifics on how delay affects individual game actions, particularly since game clients can sometimes act on user input immediately, resulting in inconsistent game state between client and server instead of having the input feel laggy. Moreover, commercial games often deploy delay compensation techniques which further obfuscate the true effects of delay on user actions.

Other related work has looked at the effects of local delay on user input in games, including models of user performance [25, 31, 34]. Some work on moving target selection has suggested an exponential model of performance with delay and target speed [12], but has done so only for targets with constant velocity. In many games, targets move with complicated motion parameters, using force-based physics for acceleration and velocity and abrupt target directional changes. For example, vehicles in most first-person shooter games, such as *Halo* (Microsoft Studios, 2001-2019)

[1]https://en.wikipedia.org/wiki/Cloud_gaming.

or *Battlefield* (Electronic Arts, 2002-2018), and players in sports games such as *Madden NFL* (EA Sports, 1988-2019) or *Mario Strikers Charged* (Nintendo, 2016), use force-based physics for movement.

We design and implement a game that isolates the fundamental action of selecting a realistically moving target with a mouse, controlling: a) the delay between the user input and the rendered action, and b) the target motion parameters – specifically, turn frequency and turn angle. The game records the elapsed time when the user first clicks on the target and the distance from the mouse cursor when the mouse button is clicked. We deploy our custom game in a user study with 56 participants, using delays of 50 to 300 milliseconds, and target turn intervals of 0 to 150 milliseconds and turn angles from 0 to 360 degrees.

Analysis of the results shows user accuracy (measured by the distance from the cursor to a target when the mouse is clicked) and user selection time (measured by the elapsed time to select the target) degrade linearly over the range of delays tested. Accuracy was lowest when targets turned less – higher turn frequency actually made targets easier to hit, since more frequent turning decreases average target speed. However, selection times increased with higher turn frequencies since target movements are less predictable.

The results presented in Section 4 were first published in the ACM Multimedia Systems Conference (MMSys) [11]. This paper adds additional comparative analysis – we revisit seven previous papers that study target selection with delay, both to determine commonalities across studies and to compare our results to earlier findings. Our analysis investigates both stationary and moving targets, and identifies research directions for further studies of delay and games.

The rest of this paper is organized as follows: Section 2 describes work on user input modeling, network games, and game actions related to our work; Section 3 describes our methodology, including developing our game and conducting a user study; Section 4 analyzes the results from the user study; Section 5 re-analyzes results from previous studies and provides a cross-study comparison; and Section 6 summarizes our conclusions; and Section 7 presents possible future work.

2 RELATED WORK

Our research is built on three main areas: models of user input (Section 2.1), impact of delay on network games (Section 2.2), and impact of delay on individual game actions (Section 2.4). Note that there are seven additional papers that are also relevant related works that are re-analyzed and discussed at greater length in the "Comparative Analysis" section (Section 5).

2.1 Models of User Input

Fitts' law is a seminal work in human-computer interaction and ergonomics that describes the time to select a stationary target based on the target distance and target width [14]. Fitts' law has been shown to be applicable to a variety of conditions (e.g., many different task settings [24]) and input devices (e.g., manual control devices as well as eye gaze [40]); it has also been extended to two dimensions [27], making it suitable for modeling target selection with a pointing device [38]; however, Fitts' aw by itself accounts for neither moving targets nor delay.

Jagacinski et al. [21], Hajri et al. [17], and Hoffmann [18] extended Fitts' law to moving targets, adding target speed to the model; however, model fit and validation was done with very few users. Hoffman [19] revised Fitts' law to consider delay, with experiments showing users have two types of responses in the presence of delay: (1) *move-and-wait* where a user provides input, then stops and waits for it to occur, repeating as necessary (for more than 700 milliseconds of delay); and (2) *continuous* where a user provides continuous input in the presence of delay, adjusting as necessary without stopping (for less than 700 milliseconds of delay). Jota et al. [23] studied the impact of delay on target selection and dragging with touch devices.

In general, the prior work in user input models does not account for delay and target objects that move with realistic motion, nor have prior input models been validated with a larger number of users or a wide range of movement conditions.

2.2 Network Games

There have been numerous studies exploring the effects of delay on traditional network games.

Quax et al. [33] studied the effect of delay on a first person shooter (FPS) game, *Unreal Tournament 2003* (Epic Games, 2002). Subjective and objective measurements showed that users are influenced by even small amounts of delay. Amin et al. [2] studied the effects of delay on player experience for another FPS game, *Call of Duty Modern Warfare 2* (Activision, 2009), focusing on network congestion. Their results suggest that delay jitter as low as 100 ms can lower Mean Opinion Score below 3 (a typical acceptability threshold), correlated with player skill.

Fritcsh et al. [16] examined delay and player performance in a massively multiplayer online role-playing game (MMORPG), *Everquest II* (Sony, 2004). User studies with controlled amounts of delay showed EverQuest's in-game strategies for coping with delay provide smooth gameplay even at delays as high as 1250 milliseconds. Chen et al. [7] analyzed network traces from another MMORPG, *Shenzhou Online* (UserJoy Technology, 2004), inferring network congestion levels that lead to players quitting the game. They confirmed a pronounced correlation between game session times and minimum delay the players experience.

Broadly, all these approaches treat an existing game as a "black box" in that individual game actions are not studied in isolation. While useful for a general understanding of the effects of delay on network games, game design and game engines may obfuscate important system details, making it unclear how the findings relate to specific player actions. Moreover, network delay in traditional online games often manifests itself as an inconsistent state on the client and the server since the client can act on player input immediately instead of having the player's input wait for a round-trip time to the server.

2.3 Cloud-based Games

There is also research, albeit less, on the effects of delay in cloud-based games. Unlike in traditional network games where a client can potentially act on user input immediately, in cloud-based games all user input is delayed by at least a round-trip time to the server.

Chen et al. [8] discussed the effects of network delay, packet loss, and bitrate on frame rates and graphics quality for the cloud-based game systems OnLive[2] and StreamMyGame.[3] The authors did not explicitly measure player performance with delay.

Other researchers have measured players in cloud-based gaming scenarios. Jarschel et al. [22] measured quality of experience in an emulated cloud-based game system. Claypool and Finkel [13] presented the results of two user studies that measured the objective and subjective effects of delay on cloud-based games, showing that both quality of experience and user performance degrade linearly with delay. Sackl et al. [36] analyzed the relationships between delay and player experience for cloud-based gaming, showing even small delays have an impact on experience.

Cloud-based game research is closer to our work than traditional network game research since the delays studied – player input to rendered output – closely match the manifestation of delays in our work. However, these papers did not analyze individual game actions with delay.

[2]https://en.wikipedia.org/wiki/OnLive.
[3]https://en.wikipedia.org/wiki/StreamMyGame.

2.4 Game Actions

An alternative approach to using a full game in a user study is to explore the effects of delay on individual (aka *atomic*) game actions.

Claypool and Claypool [10] presented a general framework describing delay and game actions that includes *precision* – the accuracy required to complete the action successfully, and *deadline* – the time required to achieve the final outcome of the action. While helpful for explaining the effects of delay on games, the precision-deadline framework does not quantify the many possible parameters that make up game actions (e.g., dodge frequency).

Raaen and Eg [34] conducted experiments with a simple button and dial interface, letting users adjust delay based on their perceptions. They found users are capable of perceiving even low amounts of delay (around 66 milliseconds).

Long and Gutwin [25] studied the effects of delay on intercepting a moving target. They found target speed directly affects the impact of delay, with fast targets affected by delays as low as 50 ms but slower targets resilient to delays as high as 150 ms.

Pavloyvych and Stuerzlinger [32] and Pavloyvych and Gutwin [31] studied target selection and following for objects moving along Lissajous curves (smooth curves, with varying turns within the curve). They found tracking errors increase quickly for delays over 110 ms but the effects of target velocity on errors is close to linear.

In general, while these approaches have helped understand delay and fundamental game actions, they generally used only basic parameters for target movements (e.g., targets with constant velocity). In many games, target objects have complicated movement paths that depend upon how often and how sharply they turn, governed by forces applied in the intended direction.

3 METHODOLOGY

To study moving target selection parameters with delay, we: (1) develop a simple game (called *Juke!*) that enables study of target selection with controlled delay and target motion using realistic motion parameters (Section 3.1); (2) conduct a user study to evaluate the impact of target motion parameters with delay (Section 3.2); (3) analyze the core results of the user study (Section 4); and (4) compare the results with those in previously published papers (Section 5).

3.1 Juke!

Our custom game is called *Juke!*, depicted in Figure 1. *Juke!* allows study of a single user action, target selection, with controlled amounts of delay and controlled motion parameters for the target. Unlike previous studies (see Section 2.4), *Juke!* uses complicated target motion common to many games, where target movement is governed by force-based physics (e.g., acceleration) and targets turn sharply (i.e., jink). In *Juke!*, the user proceeds through a series of short rounds, where in each round the player moves the mouse cursor (a blue '+') and attempts to select the target (a red ball) as quickly and accurately as possible. The player's game progress to completion is displayed in the top left corner. The user's score is displayed in the top right corner, and represents a running total of the distance of the cursor from the target when clicked (lower is better).

Upon starting the game, users are instructed to select the target as quickly and as accurately as possible via a dialog box that states "Click the green circle, then click near the target as quickly and accurately as possible. And repeat!".

The action studied in *Juke!* – selection of a 2D moving target with a mouse – is common to many PC game genres. Examples include: 1) top-down shooters (e.g., Figure 2(a) – *Nuclear Throne*, Vlambeer, 2015) where a player aims a projectile by moving the cursor to the intended moving monster; 2) first person shooters (FPS) (e.g., Figure 2(b) – *Call of Duty*, Activision, 2003) where a

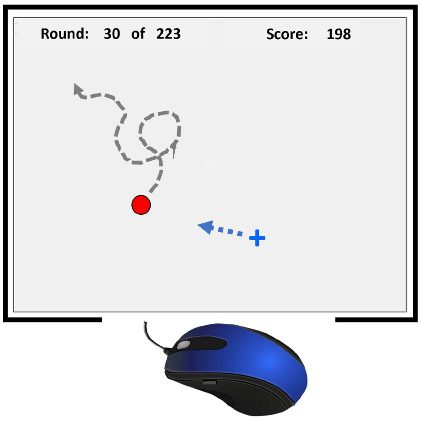


Fig. 1. *Juke!* Users move the cursor (blue cross) with a mouse and click on a moving target (red circle). The game adds delay to the user input (both mouse movement and mouse clicking) and controls the target jink frequency and jink angle using force-based movement.

(a) *Nuclear Throne* (Vlambeer, 2015), the shotgun reticle is moved with a mouse, selecting moving targets.

(b) *Call of Duty* (Activision, 2003), the world is panned with a mouse, selecting moving targets with the sniper reticle.

(c) *League of Legends* (Riot Games, 2009), the skill shot's direction is moved with a mouse, selecting moving targets.

Fig. 2. Selection of a 2D moving target with a mouse is common to many PC game genres.

Table 1. Definition of Terms

Term	Definition
force-based physics	constant acceleration in target's direction
jink/turn	change in target direction
jink interval	time between turns
jink angle	maximum angle range for turn

player uses the mouse to pan the game world to align a reticle over a moving opponent; and 3) multiplayer online battle arenas (MOBAs) (e.g., Figure 2(c) – *League of Legends*, Riot Games, 2009) where a player moves a skill shot indicator to aim at a moving opponent with a spell.

Juke! is written in Python using `tkinter`, and runs in full screen mode at 1080p resolution (1920x1080 pixels) and 33 f/s. Table 1 provides the definition of terms used in describing *Juke!* behavior. The user begins each round by clicking a green circle in the middle of the screen. This situates the user's mouse cursor at the same starting location each round. Upon clicking, the green circle disappears and a red target appears at a random location a short distance from the center of the screen.

Table 2. User Study Parameters

Milliseconds:	Degrees:	Milliseconds:
30, 90, 150	0, 90, 360	0, 62.5, 125, 250
(a) Jink Intervals.	(b) Jink Angles.	(c) Added Delays.

The target moves with force-based physics [6], applying an acceleration in the target's intended direction, with a limit on the maximum speed. The intended direction is adjusted based on how often the target jinks and how sharply the target changes direction when jinking. The direction of the target is initially randomly chosen.

After that, the direction changes every jink interval, adjusting the direction in an amount randomly chosen in the jink angle range (effectively, the target can turn at most $\frac{1}{2}$ the angle range to the left or $\frac{1}{2}$ the angle range to the right). A force of 5 pixels/s^2 is applied in the target's direction, added to the velocity. The magnitude of the velocity (maximum speed) is limited to 50 pixels/s. The velocity is added to the target's location.

The jink interval and jink angle parameters are shown in Table 2(a) and Table 2(b), respectively. The jink interval is the time between turns. The jink angle is the maximum angle range over which the target can turn where each turn angle is chosen randomly over this range. The actual turn angle chosen can affect the speed, since turning back in the opposite direction has a braking (decelerating) effect. Effectively, these parameters provide for a range of target movements that might be experienced in shooting/combat games – e.g., fast, predictable targets (jink angle 0, jink interval 0), medium-speed, less-predictable targets (jink angle 90, jink interval 90), slow, unpredictable targets (jink angle 360, jink interval 30).

The game adds a controlled amount of delay selected from the set in Table 2(c). The delay is added to all user input (mouse movement and clicks) for the duration of the round. Currently, only fixed amounts of delay are added (i.e., no jitter) because input delay usually has much lower variance than does network delay since the former comes from hardware processing and the latter primarily from variable amounts of competing traffic. However, studying the effects of delay jitter is included in our future work in Section 6.

Each jink interval, angle & delay combination appears 5 times, but the entire set of combinations is shuffled into a different random order for each user. A round ends when the user clicks the mouse or when the target moves off the edge of the screen.

The first three rounds are practice rounds, with no added delay and slow (half acceleration) targets, a jink interval of 150 ms, and a jink angle of 90 degrees. After the practice rounds, the user plays through the 36 jink rounds (4 delays x 3 intervals x 3 angles), 4 no-turn rounds (4 delays), and 1 stationary target (4 delays), 5 times each, for a total of $3 + (36 + 4 + 4) \times 5 = 223$ rounds, all shuffled randomly. Between rounds, the user is free to pause as long as necessary to rest/regain concentration. Playing through the entire set of rounds typically takes less than 15 minutes.

3.2 User Study

Our user study was conducted in windowless but brightly lit computer lab with four workstations arranged around the perimeter of the room (facing the walls); the experimenter sat in the center of the room, which allowed visual oversight of all four systems. The lab computers had Intel i7-4790 4 GHz processors, 16 GB of RAM, and NVidia GeForce GTX 960 graphics cards and ran Microsoft Windows 10. The PC monitors were Dell U2412M 24″ LCDs displaying 1920x1200 pixels @ 59 Hz.

Study participants were solicited via University email and a system whereby Psychology students participate in user studies to obtain class credit.[4] Additional incentives were a raffle for a $25 gift card and a $25 gift card for the user with the best score. Participants from an author's class also received 1 extra point on an exam.

Upon arrival in the lab, users were read a script about the study and asked to sign an IRB consent form. Then users were encouraged to adjust the computer chair height and monitor angle/tilt so as to be comfortably looking at the center of the screen. Users were told they could use the mouse with whichever hand was preferred.

Next, users completed a demographics and gaming experience survey coded using Qualtrics.[5] Lastly, the game and incentives were described and the game launched.

Users immediately started playing. As described above, the first three rounds had no added delay and slow targets for practice and the results were not recorded. Users then played through all 5x shuffled combinations of jink interval, angle & delay (Table 2(a), Table 2(b), and Table 2(c)).

3.3 Local System (Base) Delay

Note, the delays in Table 2(c) added by *Juke!* are in addition to any delays inherent in the base computer system. Since such base delays have been shown to be significant [20, 35], we measured the base delay in *Juke!* on our lab computers using a Blur-busters type technique.[6]

An LED on a breadboard was connected via a wire to a mouse so that the LED lit when the mouse button was clicked. A high frame rate camera (Casio EX-ZR100) filmed the player at 1000 f/s, capturing the moment the mouse was clicked. By manually examining the video frames, we subtracted the frame number when the light appeared (mouse was clicked) from the frame number when the display was updated based on the click, giving the base delay (in milliseconds).

The measurement method was repeated 5 times, resulting in base delay values of 50, 49, 52, 53 and 48 milliseconds. Hence, 50 milliseconds is added to all delay analysis.

4 ANALYSIS

This section first summarizes participant demographics (Section 4.1), then presents the core results – user performance in the presence of delay (Section 4.2).

4.1 Demographics

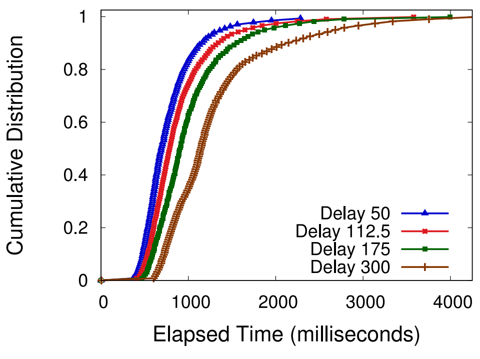
The study had fifty-six participants, aged 17-26 years (mean and median 20). Sixteen users were female and 40 male. Fifty-two were right-handed, 3 left-handed and 1 ambidextrous. User self-rating as a PC gamer (scale 1-low to 5-high) had a mean of 3.5, with a slight skew towards having "high ability". Half of the users played 6 or more hours of computer games per week. Most users majored in Robotics Engineering, Computer Science, or Game Development.

4.2 User Performance

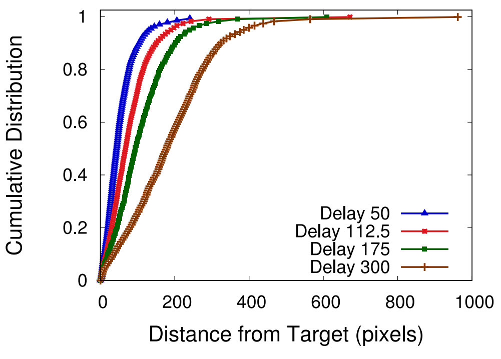
There are two primary metrics for user performance: a) *selection time* – the elapsed time from the start of the round until the mouse button is clicked; and b) *accuracy* – the distance from the mouse cursor to the target when the mouse button is clicked. Note, there are also rounds where the user fails to click the mouse before the target reached the edge of the screen (less than 5% of the time). These rounds are removed from analysis in this section, but are analyzed in Section 5.

Figure 3 shows graphs of cumulative distribution functions for selection time and accuracy. For the left graph (selection time), the x-axis is the elapsed time when the mouse button is clicked, and

[4]http://wpi.sona-systems.com/.
[5]https://www.qualtrics.com/.
[6]http://www.blurbusters.com/gsync/preview2/.

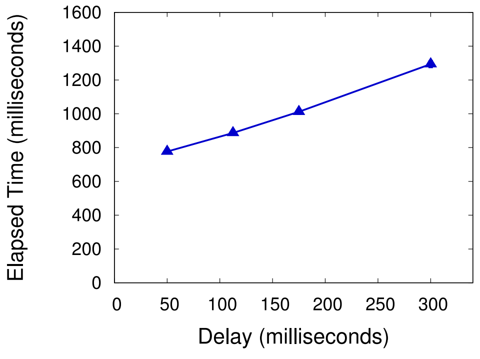


(a) User selection time.

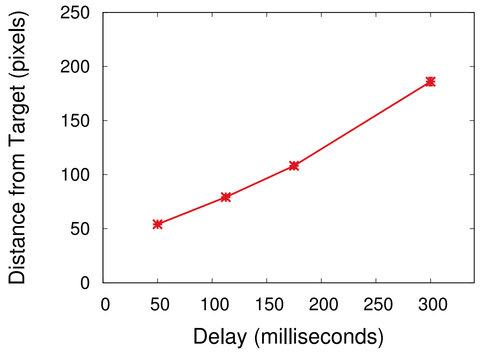


(b) User accuracy.

Fig. 3. Cumulative distribution functions, clustered by delay (milliseconds).



(a) Selection time.

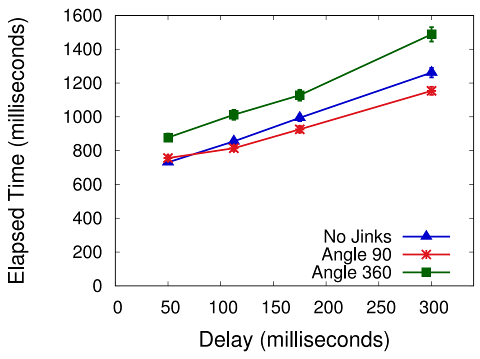


(b) Accuracy.

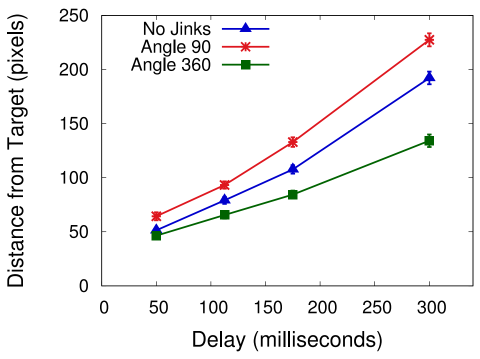
Fig. 4. User performance averaged across all users.

for the right graph (accuracy), the x-axis is the distance (in pixels) from the mouse to the target when the mouse button is clicked. For both graphs, the y-axis is the cumulative distribution. For both graphs, there is a clear separation of the CDF lines with delay, with the distributions for the lower delays towards the left (lower) and the distributions for the higher delays towards the right (higher). All CDFs have a heavy tail, with the top 5% of the distribution significantly higher than the body. The minimum elapsed times are around a half-second, but can be as high as about 4 seconds.

Figure 4 depicts graphs of selection time and accuracy versus delay. For the left graph (selection time), the y-axis is the elapsed time when the mouse button is clicked, and for the right graph (accuracy), the y-axis is the distance (in pixels) from the mouse to the target when the mouse button is clicked. For both graphs, the x-axis is the total input delay (added delay + base delay). Each point is the mean time for all users for that delay, shown with a 95% confidence interval (note, for this graph and others, the intervals are difficult to see since they are so small). Overall, there is a linear increase in selection time, and a linear decrease in accuracy, with increasing delay over the range of delays tested. This is in contrast to our previous work [12] that showed an exponential relationship with delay, albeit the ranges of delays in the previous study were larger (maximum 450 milliseconds). We carry out further comparisons regarding this issue in Section 5 below.



(a) Selection time.



(b) Accuracy.

Fig. 5. User performance averaged across all users, data grouped by target jink angle.

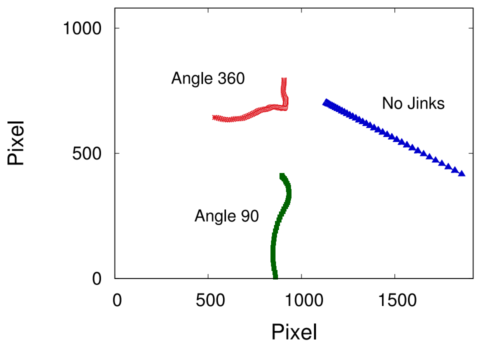
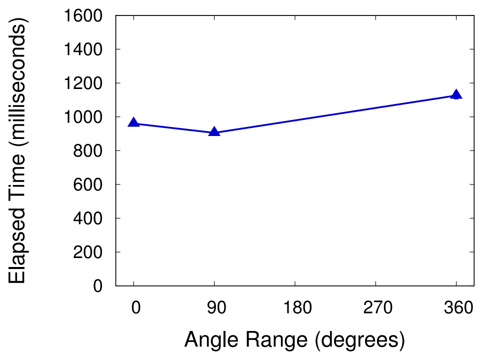


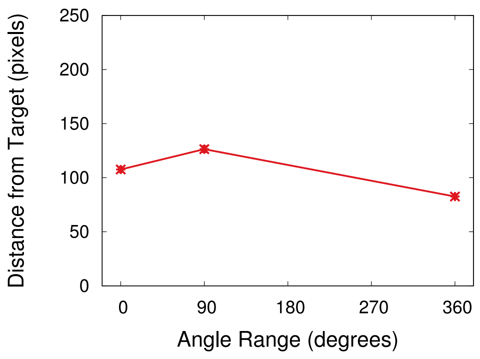
Fig. 6. Example paths traveled by 3 targets – no jinks, jink angle 90, jink angle 360.

Figure 5 graphs the same data as in Figure 4, but with data grouped by the target jink angle (the angle range used when a target turns). Both graphs have three data groups, one for each target jink angle. Each point is the mean distance for all users for that delay & angle combination, shown with a 95% confidence interval. Generally, there is a clear separation of the data groups for both graphs, indicating target jink angle affects user selection time and accuracy. For selection time, the angle 360 targets are the hardest to select, indicated by the longest times, the angle 90 targets the easiest and the no jink targets in between, but closer to the angle 90 targets. Delay impacts the elapsed time about the same for all jink angles. In contrast, for accuracy, the angle 90 targets are the hardest to select, indicated by the largest distances, the angle 360 targets the easiest and the no jink targets in between. Delay impacts the distance for all jink angles, but is most pronounced for the angle 90 targets and the trend lines diverge as delay increases.

The effects of the target jink angle are illustrated in Figure 6, which depicts the target paths for three sample targets, one for each angle tested. The point density (1 point per 30 milliseconds) on all paths indicates the relative target speeds, with points spaced further apart indicating higher target speed. When the jink angle range is large (e.g., angle range 360 degrees) the target frequently reverses direction, causing the applied force to reduce the target speed, making it easier to hit. However, the large angle range makes the target path unpredictable, causing the user to take longer to track target location and click the button. On the other hand, when the target does not turn (e.g., angle range 0 degrees), while the target moves more quickly, tracking the

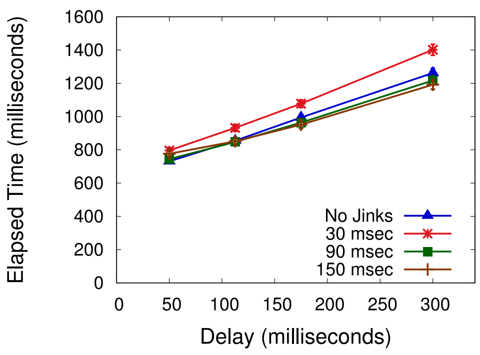


(a) Selection time.

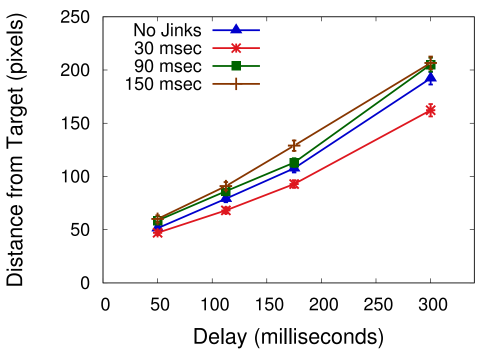


(b) Accuracy.

Fig. 7. User performance averaged across all users versus jink angle range.



(a) Selection time.



(b) Accuracy.

Fig. 8. User selection time averaged across all users, data grouped by minimum target jink interval.

target location is relatively easy since it moves in a straight line. When the target makes more gradual turns (e.g., angle range 90 degrees), a turn does not drastically slow the target, but it does make it harder to predict the location for tracking and selection.

We surmise there is one jink angle where users are most inaccurate when selecting and another where users are slowest to select the target. Figure 7 shows analysis that supports this idea. The y-axes for both graphs are as for Figure 4, but here the x-axis is the jink angle range. Data points are still means with 95% confidence intervals. For selection time, the largest angle range takes the longest to select, while the angle 0 and angle 90 angle ranges take less time to select. For accuracy, the largest angle range is actually the easiest to hit (the distance is the lowest), while the 90 degree angle range is the hardest to hit of the 3 angles tested. Future tests with more angle ranges are needed to accurately determine the shape of the curves over the full 0-360 angle range.

Figure 8 graphs the same data as in Figure 4, but the data is grouped by the jink interval (the time between target turns). The non-jink case is shown for reference. For selection time, the target that turns the most often (jink interval 30 milliseconds) takes the longest to select, while the time to select targets with higher turn intervals is similar to the time to select targets that do not turn. For accuracy, targets that turn less often are harder to hit. However, not turning at all (no jinks) makes a target harder to hit than frequent turns (jink interval 30 milliseconds), likely because of

Table 3. 3-factor ANOVA – Delay, Jink Angle, Jink Interval

Factor	F value	p
delay	2192.44	<2e-16
angle	321.63	<2e-16
interval	219.91	<2e-16
delay-angle	13.63	0.000223
delay-interval	70.02	<2e-16
angle-interval	64.17	1.24e-15

(a) User selection time.

Factor	F value	p
delay	4435.363	<2e-16
angle	264.282	<2e-16
interval	404.660	<2e-16
delay-angle	150.044	<2e-16
delay-interval	89.256	<2e-16

(b) User accuracy.

the effects of frequent turns on target speed. As for jink angle, this suggests there are different jink intervals that make a target the hardest to hit and take the longest to select, the latter somewhere above 150 milliseconds and the former somewhere less than 75 milliseconds.

The user selection time (elapsed time) and user accuracy (distance) were each analyzed using a 3-factor ANOVA.[7] A summary of the results are shown in Table 3(a) and Table 3(b), respectively. For both selection time and accuracy, there were significant main effects of delay, jink angle and jink interval, and significant interaction effects for delay-angle, and delay-interval. For selection time only, there was a significant interaction effect for angle-interval. For neither selection time nor accuracy was the interaction effect of delay-angle-interval significant.

5 COMPARATIVE ANALYSIS

This section compares the analysis in Section 4 – user performance for target selection with delay where targets move with force-based motion – with select previously published papers that also examined user performance for target selection with delay. The intent is to compare and contrast the body of work in order to help draw stronger inferences about the effects of delay on target selection where results align, and identify areas that need additional exploration when the results contradict. Section 5.1 provides a summary of the papers selected for comparison, Section 5.2 lists the key dimensions explored by the papers as related to our analysis in Section 4, and Section 5.3 presents the comparative analysis.

5.1 Selection of Papers

Papers selected for comparison (each described below) are those that examine the fundamental task of target selection with mouse input, including parameters for the target (e.g., size and motion) as well as ranges for delay. All papers did one or more within-subjects user studies with participants selected from a local university. Note that since the mouse is frequently studied and is often the input device of choice for many competitive games (e.g., first-person shooters), our comparisons are restricted to mouse input. However, other work is starting to assess differences between selection devices [26].

MacKenzie and Ware. [28] measured selection time and error rates when selecting stationary targets. Targets had fixed heights but varied in width across trials. The factors varied were delay (8.3, 25, 75, and 225 ms), target distance (96, 192, and 384 pixels), and target width (6, 12, 24 and 48 pixels). Their user study had 8 participants. Participants used a SGI Personal Iris workstation with a 19″ CRT monitor with a resolution of 1280x1024 pixels @ 60 Hz and an optical mouse with 200 dpi resolution. Note, while the system studied is old by today's standards, the authors carefully

[7]Using R, https://www.r-project.org/.

measure local delay so their results should be independent of the underlying system. Moreover, many competitive game players still use CRTs because they have lower display latency [3]. They found significant main effects for delay on selection time. Error rates increase more than 200% for a delay of 225 milliseconds compared to 8.3 milliseconds. In particular, they found a pronounced multiplicative effect between delay and Fitts' Index of Difficulty (ID) [14]. They proposed a modification to Fitts' law that incorporates delay by including an additional term, ($c \cdot LAG$), that is a weighting of delay plus the ID.

Teather et al. [39] measured selection time for stationary targets of different sizes. The factors included delay (0, 35, 75 and 225 ms) and spatial jitter in the pointer (0 or 0.3 mm). Their user study had 14 participants. Participants used a 21″ CRT monitor with a resolution of 800x600 pixels @ 120 Hz and a Microsoft optical mouse. They found the relative performance "cost" for a delay of 40 ms to be about a 15% degradation in user performance. They confirmed similar findings to MacKenzie and Ware [28] for Fitts' law computations for ID.

Pavlovych and Gutwin. [31] measured accuracy and time in tracking and selecting fixed-sized, moving targets. Targets moved using Lissajous curves (parameterized mathematical models that provide smooth curves with varying path complexity depending on the number of harmonics in the model). Delay was only added to the the mouse click and not to the mouse movement. The factors varied were delay (20, 50, 110, and 170 ms), target path complexity (1, 2 and 3 harmonics), and target speed (4, 8, 12 and 16 cycles of the Lissajous path per minute). Dependent variables were tracking and selection error and number of clicks to select the target. Their user study had 12 participants. Participants used a 24″ LCD monitor with a resolution of 1920x1080 @ 60 Hz, and a Logitech optical mouse with a 800 dpi resolution, connected via USB and sampled at 125 Hz. They found significant main effects for delay on the number of clicks for target selection and on delay for selection and tracking errors. The interaction between delay and target speed was significant as was the interaction between delay and path complexity. Target speed had a linear effect on the number of clicks needed to select a target and on selection and tracking errors. Targets moving at non-constant speed were easier to select than the same sized targets moving at a constant speed.

Friston et al. [15] measured accuracy and time in selecting stationary targets, with a focus on a low-delay system. The factors varied were delay (6, 16, 26, 26, 56 and 86 ms), target width (2.5 and 9 mm), and target distance (40 and 110 mm) for selection. Dependent variables were movement time for both selection and tracking, and error outside the path for tracking. Their user study had 30 participants. Participants used a 24″ LCD monitor with a resolution of 1920x1080 @ 144 Hz, and a Kingston Mouse-in-a-Box optical mouse connected via USB, sampled at 1000 Hz and polled directly by the system. They found that delay begins to affect performance at about 16 ms, and that the effect is non-linear. They confirmed earlier results of Fitts' law and compared their model to those in both MacKenzie and Ware [28] and Teather et al. [39].

Ivkovic et al. [20] measured selection time for fixed-sized, stationary targets and accuracy in tracking for two sizes of targets (15 and 36 mm) that moved horizontally using simple kinematics. The factors varied were delay (11, 41, 74, 114 and 164 ms), latency compensation (on or off), and target speed for tracking (240 units/s and 320 units/s). Dependent variables were task completion time for selection and time on target for tracking. Their user study had 18 participants. Participants used a 24″ LCD monitor with a resolution of 1920x1080 @ 120 Hz, and a Razer DeathAdder 3.5G optical mouse sampled at 1000 Hz. They found significant main effects for delay on completion time and tracking, with and without latency compensation. The effects of the lowest levels of delay (41 ms) was negligible, but became substantial in an approximate linear fashion for higher delays (up to 164 ms). There was a significant effect of delay on both selection and tracking task difficulty. The impact of delay on tracking was greater at lower target speeds than at higher target speeds.

Claypool, Eg and Raaen. [12] measured selection time for fixed-sized, moving targets. The authors did two separate user studies at universities in Norway and the U.S. Targets moved using simple kinematics. The factors varied were delay (20, 45, 70, 95, 120, 145, 170, 195, 220, 320, and 420 ms for one study and 100, 125, 150, 175, 200, 225, 250, 275, 300, 400, and 500 ms for the second) and target speed (150, 300 and 450 pixels/s for one study and 550, 1100 and 1550 pixels/s for the second). Dependent variables were the time and number of clicks required to select the target. In total, their user studies had 83 participants. Participants used a 24″ LCD monitor with a resolution of 1920x1080 @ 60 Hz, and an optical mouse sampled at 125 Hz. They found an exponential increase in target selection time with target speed and an exponential increase in target selection time with delay. They also found an interaction between speed and delay that showed a combination effect not realized by speed or delay alone. They proposed a model for the time to select a moving target with additive exponential terms for delay and speed and a multiplicative interaction term.

Long and Gutwin. [26] measured selection time for different sized, moving targets. Targets moved using simple kinematics. The factors studied were delay (49, 99, 149, 249, and 299 ms), target speed (0 and 1200 pixels/s) and target size (300 and 600 pixels). The dependent variables were movement time and movement errors. Their user study had 16 participants. Participants used a 28″ touch monitor with a resolution of 4500x3000 @ 60 Hz, and an MSI Interceptor DS100 optical mouse with 800 dpi and sampled at 1000 Hz. They found significant main effects for delay on selection time and that the effects of delay are exacerbated by fast target speeds.

5.2 Dimensions of Comparison

Based on parameters in the selected papers, we identify several objective measures for comparison.

Delay Range. All papers add delay to the user-provided input and the resultant output. Most papers measure and report the base delay added by the computer system under test. Base delays can vary from about 20 to 200 milliseconds [20, 35]. The delay range studied affects the ability to interpolate between delay values studies and to extrapolate beyond measured results. The amount of delay added and, hence, the total delay varies across papers from a low of 6 ms by Friston et al. [35] to a high of 500 by Claypool, Eg and Raaen [12].

Number of Users. In general, more participants in a user study provides for more statistical power. In the selected papers, all user studies were *within subjects*. The number of participants varied from 8 by MacKenzie and Ware [28] to 83 by Claypool, Eg and Raaen [12].

Target Size. Target size can impact the performance for selection tasks since the user is trying to place the mouse cursor on the target and click the button to complete the task. Target sizes are often reported in pixels, but can be converted to the actual physical size on the screen (e.g., in millimeters) based on the reported screen resolution and size. The target sizes varied from the smallest of 2.1 mm by MacKenzie and Ware [28] to the largest of 90 mm by Friston et al. [15].

Motion Type. There are four types of target motion studied in the selected papers:

(1) *None* – targets were always stationary.
(2) *Kinematics* – targets moved with immediate, constant velocity.

Table 4. Summary Table of Selected Studies on Delay and Target Selection

Publication	Year	Motion Type	Motion Parameters	Target Size	Delay Range	Performance Measures	Users	Notes
MacKenzie and Ware [28]	1993	None	N/A	2.1, 4.2, 8.4, 16.8 mm	8.3, 25, 75, 225 ms	Selection time, Error rate	8	
Teather et al. [39]	2009	None	N/A	6, 13mm	35, 75, 225 ms	Selection time	14	Also jitter
Pavlovych and Gutwin [31]	2012	Lissajous	4, 8, 12, 16 Hz 1, 2, 3 harmonics	14 mm	20, 50, 110, 170 ms	Selection time, Error rate, Distance	12	Also tracking, jitter. Click delay
Ivkovic et al. [20]	2015	None	N/A	15, 36 mm	11, 41, 74, 114, 164 ms	Selection time	18	Also tracking task
Friston et al. [15]	2016	None	N/A	25, 90 mm	6, 16, 26, 36, 56, 86 ms	Selection time	30	Both 1d and 2d tasks
Claypool, Eg and Raaen [12]	2017	Kinematics	42, 84, 126, 154, 308, 434 mm/s	28 mm	20-500 ms (22 values)	Selection time, Error rate	83	Two separate studies
Long and Gutwin [26]	2019	Kinematics	0, 158 mm/s	39.5, 79 mm	60, 110, 160, 210, 260, 310, 360, 410 ms	Selection time, Error rate	16	Also tracking, other devices
Claypool, Cockburn and Gutwin [11]	2019	Force-based	angle 0, 90, 180° jink 0, 30, 90, 150 ms	8 mm	50, 113, 175, 300 ms	Selection time, Distance	56	Performance results in Section 4.2

(3) *Lissajous* – targets moved in a complex harmonic motion governed by parametric equations.[8]
(4) *Force-based* – targets moved via force-based physics, with directional force over time determining velocity.

MacKenzie and Ware [28], Teather et al. [39], Ivkovic et al. [20], and Friston et al. [15] used only stationary targets for target selection, Claypool, Eg and Raaen [12] and Long and Gutwin [26] used kinematics, Pavlovych and Gutwin [31] used Lissajous curves, and our work [11] (Section 4 and Claypool, Cockburn and Gutwin [12]) used force-based physics.

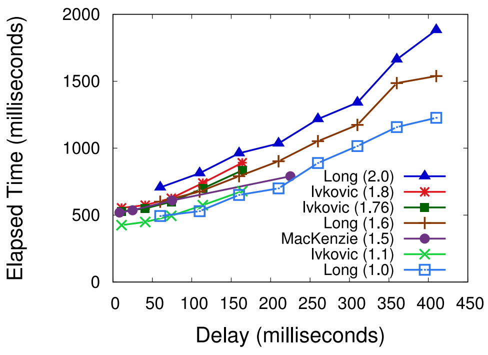
Other Dimensions

There are other dimensions in the selected works that could be compared, but are not considered here since they are not directly comparable with the results in Section 4. These can also have effects on performance, and should be considered further in future work. The factors include:

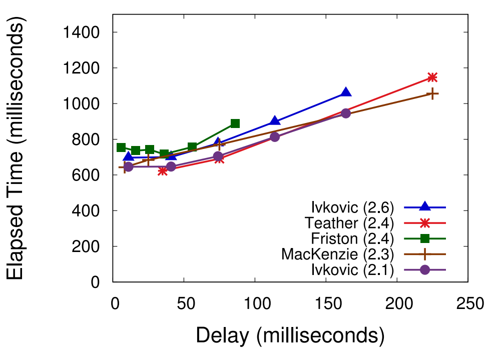
- *Input device type* – mouse or touch [26]. We have only considered input with a mouse.
- *Task type* – tracking [15, 20, 31] (users follow a moving target with the mouse) or selection (users move the mouse to the target and click). We only consider selection.
- *Jitter* – variance in delay [31]. We only consider constant delay since local (hardware-based) delays tend to vary little, with study of jitter left as future work.
- *Delay type* – whether all actions are delayed (e.g., mouse movement and mouse click) or only some of the actions (e.g., mouse click [31]). We only consider delaying all actions.

Table 4 presents a summary comparing the study parameters across the selected papers. Note, the results from the user study in our MMSys paper (Claypool, Cockburn and Gutwin [11]) are a superset of those presented in this paper (Section 4.2).

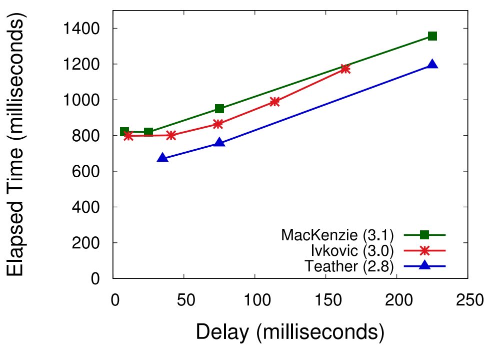
[8]https://en.wikipedia.org/wiki/Lissajous_curve.



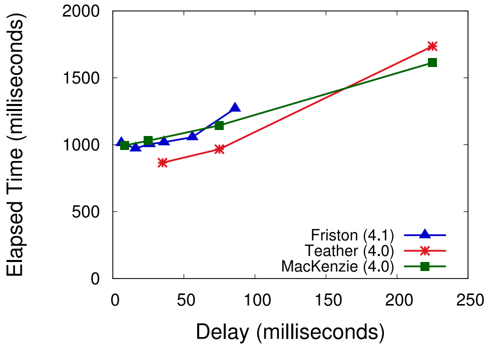
(a) Cluster 1 (IDs 1.0-2.0) R^2=0.94.



(b) Cluster 2 (IDs 2.2-2.6) R^2=0.93.



(c) Cluster 3 (IDs 2.8-3.1) R^2=0.94.



(d) Cluster 4 (IDs 4.0-4.1) R^2=0.93.

Fig. 9. Selection time versus delay, stationary targets, data grouped by ID.

5.3 Comparison of Results

5.3.1 Stationary Targets. Five of the selected papers study user performance for stationary targets: 1) MacKenzie and Ware [28], 2) Teather et al. [39], 3) Ivkovic et al. [20], 4) Friston et al. [15], and 5) Long and Gutwin [26]. They all have:

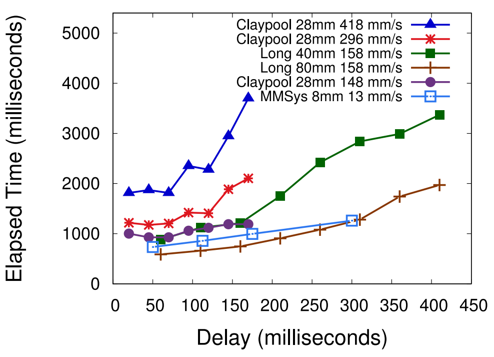
- Targets that do not move
- User performance measured by target selection time

Since the targets are stationary, we can compute the Index of Difficulty from Fitts' law [14]:

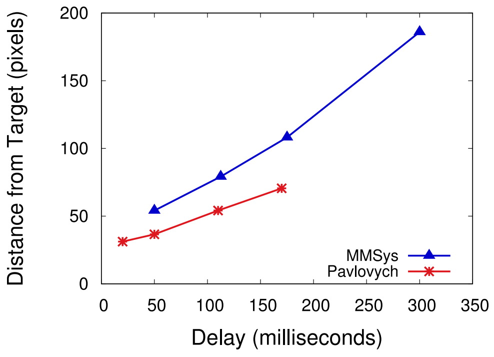
$$ID = \log_2\left(\frac{2D}{W}\right) \quad (1)$$

where D is the distance from the mouse to the center of the target and W is the width of the target. Since the target widths and distances differ somewhat for each study, the ID values are not exactly the same, either. Hence, we cluster IDs that are close. A comparison of the target selection times versus ID are depicted in Figure 9, showing four ID clusters. For each graph, the x-axis is the total delay (in ms) and the y-axis is the elapsed time to select the target (in ms). Note the x-ranges and y-ranges are slightly different in each graph. Each point is the mean elapsed time for that delay, with groups for the IDs indicated in the legend.

From the graphs, the relationship between the elapsed time to select the target and delay appears linear with delay, regardless of ID cluster. The coefficient of determination (R^2) for the lines fit to each cluster are shown in the captions of each graph.



(a) Selection time, targets moving in a line, data grouped by target speed.



(b) Distance, data grouped by motion type.

Fig. 10. Selection time and distance versus delay.

This is evidence supporting a law that relates the combination of ID and delay to selection time. From an information theoretic perspective, the ID should include a term for delay. The higher the delay, the harder it is for the user to close the control-feedback loop, requiring more information (time) to be conveyed to the user. In general, the data supports the model for time to select a target based on ID modified by a delay component developed by MacKenzie and Ware [28]:

$$T = a + (b + c \times D)ID \tag{2}$$

where a, b, and c are empirically-determined constants and D is the delay (ms). The model reduces to the standard form of Fitts' law when delay (D) is 0. The selection time at constant delay is linearly related to the ID and with fixed ID, linearly related to delay.

5.4 Moving Targets

Four of the selected papers study user performance for selecting moving targets: 1) Pavlovych and Gutwin [31] (Lissajous), 2) Claypool, Eg and Raaen [12] (kinematics), 3) Long and Gutwin [26] (kinematics); and 4) our work (Claypool, Cockburn and Gutwin [11]) (force-based).

The papers Claypool, Eg and Raaen [12], Long and Gutwin [26] and Claypool, Cockburn and Gutwin [11] (when the target does not jink) have:

- Targets moving in a straight line (kinematics)
- Performance measured by target selection time

This allows for a direct, visual comparison of results, shown in Figure 10(a). The x-axis is the delay (in ms), the y-axis is the elapsed time to select the target (in ms). Each point is the average time to select the target at the indicated delay. There are 5 groups shown, clustered by target speed (in mm per second). The data from Claypool, Cockburn and Gutwin [11] is labeled "MMSys".

Overall, the results across the three studies present a consistent picture of a higher selection time with a higher target speed. The one anomalous result is the "MMSys 13 mm/s" line in Figure 10(a) which, despite having slow targets, has a higher selection time than conditions with much higher speeds (e.g., "Long 158 mm/s"). The likely reason for this difference is that the target width for the *Juke!* study (described in Section 3) was 8 mm, which is substantially smaller than the targets in Long's study (40 mm and 80 mm), and a slow but small moving target could have a similar difficulty as a larger faster target.

Although the slower targets appear to have flatter slopes, most of the data shows a super-linear relationship with an increase in delay. The super-linear increase occurs at different delay amounts, with faster targets seeing a rise at lower delays than slower targets. This means that high-velocity targets (e.g., "Claypool 418 mm/s" in Figure 10(a)) are extremely sensitive to delay.

Only Pavlovych and Gutwin [31] and our work (Claypool, Cockburn and Gutwin [11]) have:

- Targets moving (Lissajous and force-based)
- Performance measured by distance from target

Figure 10(b) depicts a comparison of the results. The x-axis is the delay (in ms) and the y-axis is the distance from the mouse cursor to the target (in pixels). Each point is the average distance when the target was selected at the indicated delay. There are 2 groups shown, one for each data set. The data from Claypool, Cockburn and Gutwin [11] is labeled "MMSys".

Figure 10(b) shows that the two studies saw a similar increase in error distance with increasing delay, indicating that delay affects both selection time (Figure 10(a)) and selection accuracy (Figure 10(b)). These results suggest that the effects of delay are not simply due to a speed-accuracy tradeoff, since both speed and accuracy suffered with increasing delay. The difference in absolute values between the studies may be due to the other experimental factors that are collapsed in Figure 10(b), including the different target speeds and manipulations of path complexity.

Ivkovic et al. [20] showed a similar relationship between error distance and delay for their tracking task, where the error was time off target rather than distance to target. Their increase in error was not as steep as seen in Figure 10(b), but this is likely due to the difference in the measured variable (i.e., percent time off target has a natural limit whereas absolute error amount does not).

Only Pavlovych and Gutwin [31], Long and Gutwin [26], and Claypool, Eg and Raaen [12] have:

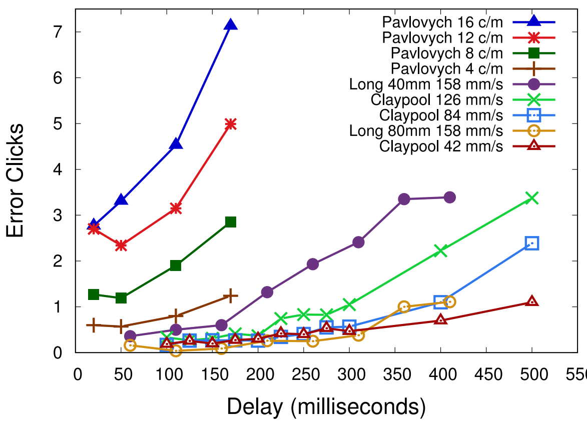
- Targets moving (kinematics and Lissajous)
- Performance measured by number of clicks

For comparison, our study with *Juke!* (labeled "MMSys") includes the number of "failures" where a user did not click the mouse before the target went off the edge of the screen.

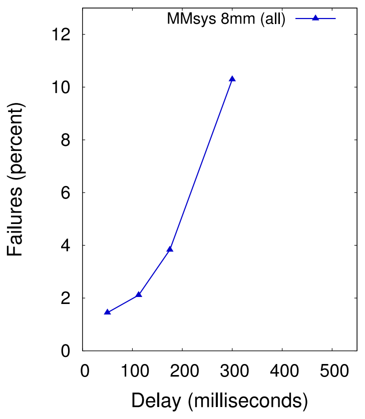
Figure 11 depicts a comparison of the results. For both graphs, the x-axis is the delay (in ms). For Figure 11(a), the y-axis is the number of extra clicks (error clicks) required to select the target, and for Figure 11(b), the y-axis is the percentage of rounds that users failed to click the mouse before the target went off the edge of the screen. For Figure 11(a), each point is the average number of error clicks for selecting the target at the indicated delay, grouped by data set.

As shown in both graphs in Figure 11, there is a clear effect across studies of delay on the number of user errors. All datasets appear to indicate a super-linear relationship between delay and errors, with the curves becoming steeper as targets become faster and more difficult to select. The Pavlovych results in Figure 11(a) also follow this trend, but are all substantially higher even at low delay amounts. We estimate the speeds of the Pavlovych targets at 38-150 mm/sec, so these higher error amounts are not simply due to higher target speeds. However, the performance difference may be accounted for by differences in the task – in the Pavlovych study, the task involved both selecting and tracking (participants were asked to click on the target and then start moving the mouse to track the target) whereas in the other studies, the task ended with selection. The longer timeframe of the Pavlovych study may have led to an increase in the number of clicks (for example, if participants re-clicked when they went out of the target during tracking).

We also considered the relationship between error and path complexity (e.g., jink angle in our study, or harmonic motions in the Lissajous curve in the Pavlovych study). Only Pavlovych and Gutwin [31] and our work (Claypool, Cockburn and Gutwin [11]) have models of target motions

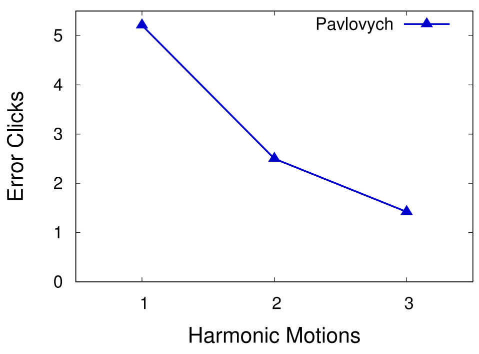


(a) Error clicks versus delay, data grouped by target speed. Targets in the Pavlovych study ranged from peak speeds of 56 mm/s (with 4 cycles per minute (cpm)) to approximately 224 mm/s (with 16 cpm).

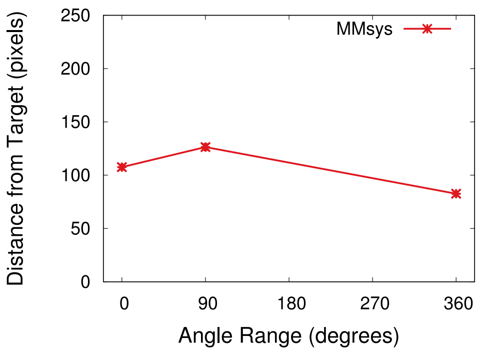


(b) Failures versus delay, *Juke!*

Fig. 11. Errors versus delay.



(a) Error clicks versus harmonic motions, Pavlovych and Gutwin [32].



(b) Distance versus jink angle, MMSys [11].

Fig. 12. User performance versus target path complexity.

with varied complexities. As noted in Section 4.2, higher jink angles did not always lead to increased error (distance), but could, in fact, decrease error. The Pavlovych study shows a similar result, where increasing the path complexity (by increasing the number of harmonics in the Lissajous curves) reduced the error. Figure 12 shows a side-by-side comparison of the two results. The Pavlovych graph on the left shows the error clicks on the y-axis versus the number of curve harmonics on the x-axis, with each point the mean number of clicks. The MMSys graph on the right is a repeat of Figure 5(b). Pavlovych et al. [31] also observed this relationship.

These findings suggest a connection between the difficulty of predicting a target's motion and the difficulty of selecting the target once the user has made a prediction about location. As delay increases, the results suggest that target speed becomes critically important for target selection. In both our MMSys [11] and the Pavlovych [31] studies, increased path complexity led to a reduction in target speed because of direction changes – and this may have made the target easier to select.

Reducing target speed during periods of high delay improves performance because it reduces the amount of error between the visible target and the actual target. In addition, the reduced error may be a cue to users that the effects of delay are reduced and users may adopt a strategy of waiting until the speed of the target decreases, and then click on the visual representation, since the low speed means that the target's true position is closer to its on-screen position.

It is important to note, however, that this phenomenon only occurs in games that use a movement model for target motion in which direction changes slow the target (and where this movement model is not applied to the motion of the mouse cursor). In games that do not use realistic physics, direction reversals do not necessarily slow the target (e.g., Mario-style games allow instantaneous direction changes with no slowdown), and in these settings we expect that increased complexity would contribute to higher selection error. A limited confirmation of this expectation is seen the tracking study by Ivkovic et al. [20] where targets reversed direction immediately and accelerated back to full speed in 140 ms; in this study, there was no improvement to tracking when the target reversed direction.

6 DISCUSSION AND CONCLUSION

Our three goals in this research were to provide a better understanding of delay for game developers, to inform the design of delay-mitigation schemes, and to construct models of the effects of delay. First, our study and comparison provide several new results that can be used by developers:

- Delay matters, and more is worse. Our survey shows that studies of delay consistently find significant effects of delay on performance, and that increasing delay leads to a super-linear decrease in performance.
- Target speed matters. Studies consistently show that player interactions with fast objects are affected increasingly by delay; network designers should explicitly consider the speed of game objects, particularly when only play-testing in low-delay conditions.
- Complex target motion is not always harder to select. Two studies showed that different path characteristics – speed and complexity – can have different effects on selection performance. The idea that complex motion paths become easier to target (when using realistic physics) is a novel finding that should be considered by network game designers.

Second, although we did not explicitly study delay compensation, our results about the universality of delay's negative effects underscore the importance of mitigating delay. In addition, our results provide insights that can assist in the design and deployment of compensation techniques:

- The relationship between object speed, delay, and performance provides designers with information about where and when a compensation technique will be most valuable – previous work has already shown that compensation is useful (e.g., "sticky" targets [20]), and our work provides information about the amount of delay where performance degrades.
- Our results also suggest that certain types of compensation could be more effective when targets are fast (e.g., time warp/rollback) because the compounding effects of delay and speed could be avoided; "world alteration" techniques [37] could also be effective by adjusting game parameters (e.g., target speed) directly.
- Our results about path complexity suggest a novel approach to latency compensation: designers could use path complexity as a delay-mitigation technique – that is, as delay increases, the game could make target paths more complex (and thus easier to select). This has the advantage of not seeming artificial to the player (unlike "sticky" targets, for example), and therefore could be better accepted by players.

Third, our results clearly show that delay should be included in models of targeting and selection.

- As discussed in Section 5.3.1, our results show a linear relationship between delay and Index of Difficulty (ID). This supports the idea that Fitts' law should be extended to include a term for delay, such as: $T = a + (b + c \times D)ID$; further work is needed to establish values for the constants.
- In addition, more specific models can be produced to capture different characteristics of object motion (e.g., such as the jink rate used in our study). A template for these specific models can be seen in Claypool et al. [12]; an important activity for future work is to produce a public dataset where researchers can deposit new study results and improve the models.

Computer games are increasingly networked, subjecting players to delays not only from the local system but also from the network and game servers, particularly for cloud-based games. This means player input, such as using a mouse to align a reticle on an opponent, is delayed between when the player clicks the mouse until the game renders the change. While past work has shown delay degrades the player experience, details on the effects of delay on moving target selection, especially parameterized by target motion, have not been fully understood. In particular, what is needed is study of motion common to many games, where targets move along complicated movement paths, governed by turn frequency and turn angle with acceleration.

This paper presents the results of a study where users played a custom game called *Juke!* that isolated target selection and controlled for delay and target motion parameters – the time between target turns and the target turn angle range – using force-based physics to move the target. Fifty-six users played about 13,000 rounds of *Juke!* with the results analyzed along the independent variables of delay, jink angle and jink time.

The analysis shows that both the distance between the mouse cursor and the moving target and the time it takes the user to select the moving target increase linearly over the range of delays tested (50-300 milliseconds), with user performance decreasing 4-fold over this delay range. The target jink angle has a lesser effect on distance (40% difference), with the hardest to hit target jink angle less than 360 degrees, and with the opposite effect on selection time. The target jink interval also has a lesser effect on distance (up to 30% difference) and selection time (up to 15% difference), with the hardest to hit target jink interval above 150 milliseconds.

In additional, seven prior publications that also studied delay and target selection – [12, 15, 20, 26, 28, 31, 39] – are re-analyzed and the findings compared to those in this paper. Comparison of results from five studies with stationary targets provides strong support for a law that relates the combination of Fitts' ID and delay to selection time. Comparison of results from three studies with moving targets are consistent in showing a higher selection time with a higher target speed, with one exception (this last suggesting that a small target size might remove the benefits of a slow target). Most of the data shows a super-linear relationship with an increase in delay, with faster targets being especially sensitive to delay. Comparison of results from two studies that measure selection error confirm both speed and accuracy suffer with increase in delay. All datasets appear to indicate a super-linear relationship between delay and error clicks, with the curves becoming steeper as targets become faster and more difficult to select. Comparison of results from two studies that vary target path complexity show increasing path complexity may improve user performance since the additional complexity can slow and reduce the distance covered by the target.

7 FUTURE WORK

As noted in Section 4, in order to refine understanding of the effects of target turn angle and frequency, future work could provide for additional testing of more angles between 90 degrees and 360 degrees, as well as turn intervals greater than 150 milliseconds. Finding the threshold for tolerable delay for different movement parameters could also be of interest.

While our study controlled for target turn angle, turn interval and a maximum target speed, the acceleration force applied was constant. The magnitude of the force affects the target speed over time, particularly in the presence of turns. Future work could explore a range of forces, perhaps coupled with different maximum speeds.

Our comparative analysis suggest a closer examination of target speed and target size with delay may be warranted. Similarly, the impact of acceleration (and deceleration) on the user performance in target selection could be explored.

While local delays are fairly constant, network delays can vary considerably with bursts of traffic. Thus, future work might examine the effects of delay jitter on target selection, particularly relevant for cloud-based games.

Since another common game action is steering – navigating an avatar/vehicle down a virtual path – future work might study steering in the presence of delay. Although initial studies have been carried out in this area (e.g., Friston et al. [15]), further work is needed to develop a revised steering law [1] (a predictive model of human movement that describes the time required to navigate, or steer, through a 2-dimensional tunnel) in the presence of delay.

Last, but not least, all studies, ours as well as the seven examined for comparative analysis, have sample pools drawn from university students. Future work could conduct user studies over a broader age range, perhaps considering additional demographic breadth as well.

ACKNOWLEDGMENTS

Thanks to Chaiwat Ekkaewnumchai and Bhon Bunnag for development of *Juke!* and conducting the user studies.

REFERENCES

[1] Johnny Accot and Shumin Zhai. 1997. Beyond Fitts' law: Models for trajectory-based HCI tasks. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. Atlanta, Georgia, USA, 295–302.

[2] Rahul Amin, France Jackson, Juan E. Gilbert, Jim Martin, and Terry Shaw. 2013. Assessing the impact of latency and jitter on the perceived quality of Call of Duty Modern Warfare 2. In *Proceedings of HCI – Users and Contexts of Use*. Las Vegas, NV, USA, 97–106.

[3] Justin Andress. [n.d.]. Why 'Super Smash Bros. Melee' Pros Play on Ancient TVs. https://tinyurl.com/twhkkdb.

[4] 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. In *Proceedings of ACM Network and System Support for Games Workshop (NetGames)*. Portland, OG, USA.

[5] Yahn W. Bernier. 2001. Latency compensating methods in client/server in-game protocol design and optimization. In *Proceedings of the Game Developers Conference*. San Francisco, CA, USA.

[6] M. Buckland. 2010. *Programming Game AI by Example*. Jones & Bartlett.

[7] K. Chen, P. Haung, G. Wang, C. Huang, and C. Lee. 2006. On the sensitivity of online game playing time to network QoS. In *Proceedings of IEEE Infocom*. Barcelona, Spain.

[8] Kuan-Ta Chen, Yu-Chun Chang, Hwai-Jung Hsu, De-Yu Chen, Chun-Ying Huang, and Cheng-Hsin Hsu. 2014. On the quality of service of cloud gaming systems. *IEEE Transactions on Multimedia* 26, 2 (Feb. 2014).

[9] Mark Claypool. 2005. The effect of latency on user performance in real-time strategy games. *Elsevier Computer Networks, Special Issue on Networking Issues in Entertainment Computing* 49, 1 (Sept. 2005), 52–70.

[10] Mark Claypool and Kajal Claypool. 2006. Latency and player actions in online games. *Commun. ACM* 49, 11 (Nov. 2006).

[11] Mark Claypool, Andy Cockburn, and Carl Gutwin. 2019. Game input with delay - moving target selection parameters. In *Proceedings of the 10th Annual ACM Multimedia Systems Conference (MMSys)*. Amherst, MA, USA.

[12] Mark Claypool, Ragnhild Eg, and Kjetil Raaen. 2017. Modeling user performance for moving target selection with a delayed mouse. In *Proceedings of the 23rd International Conference on MultiMedia Modeling (MMM)*. Reykjavik, Iceland.

[13] Mark Claypool and David Finkel. 2014. The effects of latency on player performance in cloud-based games. In *Proceedings of the 13th ACM Network and System Support for Games (NetGames)*. Nagoya, Japan.

[14] Paul M. Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology* 47, 6 (June 1954), 381–391.

[15] Sebastian Friston, Per Karlström, and Anthony Steed. 2016. The effects of low latency on pointing and steering tasks. *IEEE Transactions on Visualization and Computer Graphics* 22, 5 (May 2016), 1605–1615.

[16] Tobias Fritsch, Hartmut Ritter, and Jochen H. Schiller. 2005. The effect of latency and network limitations on MMORPGs: A field study of everquest 2. In *Proceedings of the 4th ACM Network and System Support for Games (NetGames)*. Hawthorne, NY, USA.

[17] Abir Al Hajri, Sidney Fels, Gregor Miller, and Michael Ilich. 2011. Moving target selection in 2D graphical user interfaces. In *Proceeding of IFIP TC Human-Computer Interaction (INTERACT)*. Lisbon, Portugal.

[18] Errol Hoffmann. 1991. Capture of moving targets: A modification of Fitts' law. *Ergonomics* 34, 2 (1991), 211–220.

[19] Errol Hoffmann. 1992. Fitts' law with transmission delay. *Ergonomics* 35, 1 (1992), 37–48.

[20] 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 ACM SIGCHI Conference on Human Factors in Computing Systems*. Seoul, Korea, 135–144.

[21] R. Jagacinski, D. Repperger, S. Ward, and M. Moran. 1980. A test of Fitts' law with moving targets. *The Journal of Human Factors and Ergonomics Society* 22, 2 (April 1980), 225–233.

[22] Michael Jarschel, Daniel Schlosser, Sven Scheuring, and Tobias Hossfeld. 2011. An evaluation of QoE in cloud gaming based on subjective tests. In *Proccedings of the Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. Seoul, Korea, 330–335.

[23] 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 ACM SIGCHI Conference on Human Factors in Computing Systems*. Paris, France.

[24] R. Kerr. 1973. Movement time in an underwater environment. *Journal of Motor Behavior* 5, 3 (1973), 175–178.

[25] M. Long and C. Gutwin. 2018. Characterizing and modeling the effects of local latency on game performance and experience. In *Proceedings of the ACM Symposium on Computer-Human Interaction in Play (CHI Play)*. Melbourne, VC, Australia.

[26] Michael Long and Carl Gutwin. 2019. Effects of local latency on game pointing devices and game pointing tasks. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*. Glasgow, Scotland, UK.

[27] I. S. MacKenzie and W. Buxton. 1992. Extending fitts' law to two-dimensional tasks. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*. Monterey, CA, USA, 219–226.

[28] I. Scott MacKenzie and Colin Ware. 1993. Lag as a determinant of human performance in interactive systems. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 6.

[29] James Nichols and Mark Claypool. 2004. The effects of latency on online madden NFL football. In *Proceedings of the 14th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. Kinsale, County Cork, Ireland.

[30] Lothar Pantel and Lars C. Wolf. 2002. On the impact of delay on real-time multiplayer games. In *Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. Miami, FL, USA.

[31] A. Pavlovych and C. Gutwin. 2012. Assessing target acquisition and tracking performance for complex moving targets in the presence of latency and jitter. In *Proceedings of the Graphics Interface Conference*. Toronto, ON, Canada.

[32] A. Pavlovych and W. Stuerzlinger. 2011. Target following performance in the presence of latency, jitter, and signal dropouts. In *Proceedings of the Graphics Interface Conference*. St. John's, NL, Canada, 33–40.

[33] 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 ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*.

[34] K. Raaen and R. Eg. 2015. Instantaneous human-computer interactions: Button causes and screen effects. In *Proceedings of the 17th HCI International Conference*. Los Angeles, CA, USA.

[35] Kjetil Raaen and Andreas Petlund. 2015. How much delay is there really in current games? *Proceedings of ACM Multimedia Systems* (March 2015).

[36] A. Sackl, R. Schatz, T. Hossfeld, F. Metzger, D. Lister, and R. Irmer. 2016. QoE management made uneasy: The case of cloud gaming. In *IEEE Conference on Communications Workshops (ICC)*. Kuala Lumpur, Malaysia.

[37] Shafiee Sabet Saeed, Steven Schmidt, Saman Zadtootaghaj, Carsten Griwodz, and Sebastian Moller. 2018. Towards applying game adaptation to decrease the impact of delay on quality of experience. In *Proceedings of IEEE International Symposium on Multimedia (ISM)*. 114–121.

[38] R. W. Soukoreff and I. S. MacKenzie. 2004. Towards a standard for pointing device evaluation – perspectives on 27 years of Fitts' law research in HCI. *Elsevier International Journal of Human-Computer Studies* 61, 6 (2004), 751–789.

[39] R. J. Teather, A. Pavlovych, W. Stuerzlinger, and I. S. MacKenzie. 2009. Effects of tracking technology, latency, and spatial jitter on object movement. In *Proceedings of the IEEE Symposium on 3D User Interfaces*. Lafayette, LA, USA, 43–50.

[40] C. Ware and H. Mikaelian. 1987. An evaluation of an eye tracker as a device for computer input. In *Proceedings of ACM Human Factors in Comp. Systems and Graphics Interface*. Toronto, Canada.

Received December 2019; revised February 2020; accepted March 2020