The Impact of Latency on Target Selection in First-Person Shooter Games

Shengmei Liu and Mark Claypool
sliu7,claypool@wpi.edu
Worcester Polytechnic Institute, Worcester, MA
USA

ABSTRACT

While target selection in a 2D space is fairly well-studied, target selection in a 3D space, such as shooting in first-person shooter (FPS) games, is not, nor are the benefits to players for many latency compensation techniques. This paper presents results from a user study that evaluates the impact of latency and latency compensation techniques on 3D target selection via a bespoke FPS shooter. Analysis of the results shows latency degrades player performance (time to select/shoot a target), with subjective opinions on Quality of Experience (QOE) following suit. Individual latency compensation techniques cannot fully overcome the effects of latency but combined techniques can, letting players perform and feel as if there is no network latency. We derive a basic analytic model for the distribution of the player selection times which can be part of a simulation of a full-range of FPS games.

CCS CONCEPTS

• **Applied computing** → **Computer games**; • **Human-centered computing** → *User studies.*

KEYWORDS

gamer, FPS, user study, lag, shooting

ACM Reference Format:
Shengmei Liu and Mark Claypool. 2023. The Impact of Latency on Target Selection in First-Person Shooter Games. In *Proceedings of the 14th ACM Multimedia Systems Conference (MMSys '23), June 7–10, 2023, Vancouver, BC, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3587819.3590977

1 INTRODUCTION

Computer games are one of the world's most popular forms of entertainment, with global sales increasing at an annual rate of 10% or more [45]. The largest esports prize pools are about $25 million USD [10], larger even than traditional sports who's prize pools range from about $2 to $20 million USD [36]. By 2023, there are expected to be about 300 million frequent viewers of esports worldwide, an increase from 173 million in 2018 [14].

MMSys '23, June 7–10, 2023, Vancouver, BC, Canada

ACM ISBN 979-8-4007-0148-1/23/06…$15.00
https://doi.org/10.1145/3587819.3590977

Among the myriad game genres available and played with esports, the first-person shooter is one of the most popular. In first-person shooter games, players take a first-person perspective of an avatar and then move and shoot targets to accomplish game goals, often playing with other people as teammates or opponents. Latency – delay between a player's input and the game responding with audible or visual output – makes first-person shooter games less responsive, degrading player performance and hurting the quality of experience. There are two main sources of latency in first-person shooter games: 1) from the local system, such as from the mouse, OS and monitor, and, 2) from the network between the client and the server. While both sources of latency affect the player, they manifest differently -– local latency lags all player input until game output, while network latency lags communication with the server. This means local latency makes game controls feel unresponsive, while network latency makes player actions resolve later by the server. Latency compensation techniques can mitigate the effects of network latency on players and may improve player performance and quality of experience [31].

There have been numerous studies on latency and commercial games [13, 15], especially network latency and first-person shooter games [1–3, 22, 25, 29, 30, 40] owing to the sensitivity of first-person shooter games to network latency and the popularity of first-person shooter games in the competitive and esports scenes. The most common approach to assess latency is via user studies, but these can be expensive and time-consuming. Moreover, given the wide-variety of games even within a single genre (e.g., consider the scope of first-person shooter games available today) it is not practical nor probably even possible to cover all current and future game configurations with user studies.

An alternate approach is to study the effects of latency on individual game actions [25, 32, 41] which has the potential to generalize to many games and even other interactive applications. For example, the study of 2D target selection can yield results that generalize to player performance in games with the same action [24]. Studies assessing the impact of latency on target selection in 2D space have varied target parameters (e.g., target size and distance) [12, 16, 34, 44] and target motion (e.g., velocity and acceleration) [6, 35, 38, 39], as well as pointing device [5, 33]. A key outcome of these approaches are analytic models that can explain and predict the effects of latency for a wide-range of games and latency conditions. But these approaches have generally fallen short in evaluating the impact of latency compensation techniques, particularly those used in first-person shooter games. Moreover, in 3D target selection, such as shooting in a FPS game, the 2D size and speed of the target on the screen changes with the position and orientation of the target in the virtual 3D space. We isolate the target selection action in a first-person shooter game and study the

impact of local latency, network latency and latency compensation techniques on player performance and quality of experience. Then, we provide models of the player selection times given the latency and latency compensation. A model of 3D selection can potentially be combined with a model of first-person navigation [25] to simulate FPS game performance over a range of game and latency conditions.

This paper presents the results of a 39-person user study where each participant plays a custom target selection game that isolates and mimics aiming and shooting in a first-person shooter game with controlled amounts of latency and latency compensation techniques. The local latencies experienced by the users range from 25 ms to 100 ms and network latencies range from 0 ms to 150 ms, covering the range of latencies players typically experience over the Internet [16, 37]. Game logs provide objective performance measures and player input actions, while surveys provide a Mean Opinion Score (MOS) questionnaire after each game round.

Analysis of the results indicates both subjective Quality of Experience (QoE) and objective player performance degrade linearly with total latency over the range of latencies studied. A 100 ms increase in latency results in about an 2 second increase in time to select (shoot a target) and a 0.4 decrease in QoE (on a 5 point scale). Both latency compensation techniques investigated – time warp and self-prediction – can improve player performance and QoE and, when applied together, can nearly completely overcome the negative effects of network latency. We derive analytic models of the distributions of the times to select that generalize with latency and latency compensation and may be useful in subsequent first-person shooter simulations.

The rest of this paper is organized as follows: Section 2 describes previous work on latency and games related to this paper; Section 3 details our methodology, including game and user study design and execution; Section 4 presents participant demographics and analysis from the user study; Section 5 discusses some of the implications of our findings; Section 6 mentions some limitations of our approach; and Section 7 summarizes our conclusions and possible future work.

2 RELATED WORK

This section describes related work in four main areas: target selection (Section 2.1), local latency and games (Section 2.2), network latency and games (Section 2.3), and latency compensation techniques (Section 2.4).

2.1 Target Selection

Target selection in games is when a player moves a pointing device to an object on the screen and then "clicks" on it (e.g., by pressing a mouse button). In most cases, selecting a target as fast as possible is desirable, but accuracy is often considered, too.

Long and Gutwin [32] study the effects of latency on selecting a moving target. They find target speed directly affects the impact of latency, with selecting fast targets affected by latency as low as 50 ms, but selecting slower targets resilient to latency as high as 150 ms. Their follow-on study [33] measures selection times for different sized moving targets. They find that the effects of latency on selection are exacerbated by fast target speeds. Claypool et al. [6] investigate selecting a moving target with a mouse in the presence of latency. Their analysis shows target selection times are impacted exponentially by latency and target speed for constant-velocity targets. Janzen and Teather [17] conduct a study with 12 users playing a target selection game with frame rates from 15 to 60 f/s and latencies from 0 to 100 ms. Their work reveals low frame rates have a significant performance cost, and that in the lowest frame rate conditions latency does not significantly affect performance. Liu et al. [24] combine datasets [26] to model target selection times based on target size, target speed and player skill.

In general, while these approaches and previous work have helped understand latency and target selection, they have not derived models of the distributions of the selection times. Moreover, the previous work is for target selection in a 2D environment where the target sizes stay the same over the entire round. In contrast, for 3D target selection, the focus of our work, the 2D size of the target changes over the round as the target (e.g., a moving opponent avatar) moves further and closer.

2.2 Local Latency and Games

Understanding the effects of local latency on games and game-like actions can help motivate the design and development of end-host systems that benefit game players and other users doing interactive tasks.

Ivkovic et al. [16] find significant effects for local latency on target tracking and acquisition tasks, both with and without aim assistance, and with a greater effect for higher target speeds. Investigating using a full game, Liu et al. [29] measure performance and quality of experienced for skilled players in *Counter Strike: Global Offensive* (Valve, 2012) showing degradations with local latency – both subjective and objective scores decrease about 20% with a 100 ms increase in local latency. Since for cloud-based game streaming, local latency and network latency manifest similarly in their delay from player input to visual output, Claypool and Finkel [7] find both quality of experience and user performance degrade linearly with an increase in latency for cloud-based game streaming systems.

While useful for understanding and even modeling the effects of local latencies on some gaming tasks and even a first-person game, the impact of local latency on first-person target selection has not been isolated and studied.

2.3 Network Latency and Games

Understanding the effects of network latency on games can help motivate deployment or development of latency compensation techniques [4, 31] to improve user experience when interacting with other players over a network. This section concentrates on first-person shooter games.

Dick et al. [8] show via a survey that players generally think about 120 ms is the maximum tolerable latency for a network game, regardless of game genre, but their user study shows players find 150 ms acceptable for the two first-person shooter games tested. Amin et al [1] find player expertise defines and determines the sensitivity to latency for the first-person shooter game *Call of Duty* (Activision, 2003), with competitive gamers more adept at compensating for impaired conditions. Armitage et al. [2] estimate the latency tolerance threshold for the first-person shooter game *Quake*

3 (id, 1999) to be about 150-180 ms. Quax et al. [40] show players for the first-person shooter game *Unreal Tournament 2003* (Epic, 2003) that latency and latency jitter under 100 ms can degrade player performance and quality of experience. Liu et al. [28, 30] show that while local latency in the first-person shooter game *CS:GO* (Valve, 2012) has higher impact than network latency on players, reductions in network latency also benefit player performance and quality of experience.

While such work has been instrumental in better understanding the effects of network latency on first-person shooter games, their results are for specific games and may not generalize, nor do they provide an opportunity to study, much less model, the effects of latency on 3D target selection.

2.4 Latency compensation Techniques

There are numerous software techniques designed to compensate for the effects of latency on game players [4, 31]. Techniques common to first-person shooter games include: *self-prediction* where the client predicts game state based on player input before getting confirmation from the server; *extrapolation* (e.g., dead reckoning) and *interpolation* where a client predicts states for objects controlled by the server or other players based on past state; and *time warp* where the server rolls back the game state to when the player action occurred on the client, applies the action, then rolls the game state forward to the current time.

Lee and Chang [22] evaluate how interpolation in CS:GO improves player accuracy, and their follow-on work [21] suggests keeping network latencies below 250 milliseconds when using CS:GO latency compensation. Liu et al. [30] evaluate latency compensation for CS:GO, showing it can significantly improve player performance with network latency.

There is little formal evaluation of individual latency compensation techniques for first-person shooters in general, nor of the benefit of latency compensation to 3D target selection. Our paper takes steps to start to address these shortcomings.

3 METHODOLOGY

To assess the effects of latency on 3D target selection in a first-person game, we built a custom game that isolates the shooting action, implemented two latency compensation techniques, added controlled amounts of latency, recruited participants, and measured player performance and quality of experience.

3.1 3D Target Selection Game

We designed and implemented a custom game in Unity[1] that isolated the action of 3D target selection in a first-person shooter-type setting. In the game, the player stays at a fixed position at a corner of the map and can rotate to change orientation and aim, but cannot move or otherwise change positions. The opponent is a bot that spawns at a random location moving along one of three possible linear paths in the field of view and changes directions to avoid being hit. The opponent's 3d avatar size is fixed throughout, but the 2d size as seen by the player on the screen changes as the bot moves towards or away from the player. Specifically, movements away from the player result in a decreasing 2d target size while movements towards the player are an increase. While the well-known Fitts' Law [11] provides for a relationship between target selection time and distance to move the selection device (here, the reticle), it applies to fixed-size targets and does not consider targets that change size (e.g., 2d sizes changes as the 3d avatar moves). The 2d sizes of the opponent bot recorded in our custom game ranges from 1.6 cm^2 where the bot is far away and appears small on screen to 2105 cm^2 where the bot is close and takes most part of player's sight.

Figure 1: 3D target selection game screenshot. The player tries to shoot the avatar by moving the reticle (in red) to the avatar and clicking the mouse. The avatar moves and jumps to avoid being shot. The "health" bar shows player health remaining and decreases with time. "Score" shows the cumulative points over all rounds – the faster the kills, the higher the score. "Kill" shows the cumulative kills over all rounds. "Time" shows the seconds left for the current round.

To mimic opponent movement in first-person shooter games, we extracted player movements patterns from data from a previous user study on Counter Strike: Global Offensive (CS:GO, Valve, 2012) [29]. Specifically, we obtain the frequency of direction changes and jumps and the distribution of intervals between the same and use these values as the basis for our bot. From the data, the bot changes direction randomly every 3.0 - 8.7 seconds with a standard deviation of 1.4 seconds and jumps randomly every 1.4 - 10.0 seconds with a standard deviation of 2.4 seconds. The player tries to shoot the bot on the screen as fast as possible using a pistol with unlimited ammunition and a firing rate of 1 shot every 250 ms. It takes two hits to kill the bot. While the bot cannot shoot or otherwise damage the player, in order to provide some urgency for the player to shoot the bot quickly, the player's health is shown to decrease the longer the player takes to kill the bot.

The update rate for the game engine is fixed at 50 Hz. Each frame, the game logs the ongoing score for the player, the position of the enemy, the 2D distance of the reticle to the avatar, and the 3D distance between the player's position and the bot's position. The game logs every shot as a hit or miss with corresponding timestamps.

A game round is over after the bot is killed (2 hits) or after 40 seconds, whichever comes first.

The game has one map – a single, square room, 36 meters in length and width, without any cover or obstacles. The player is always at a fixed location on the map in a corner. Before the round starts, the player is given a countdown timer whereupon a bot

[1] https://unity.com/

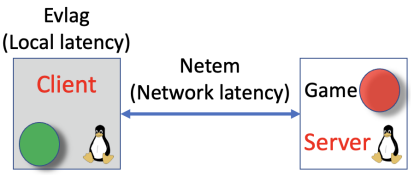


Figure 2: Client and server configuration.

spawns at one of three different locations in the field of view. Figure 1 shows a screenshot of the game where the player is aiming at the target.

The game has a client-server architecture typical of most network games where an authoritative server keeps the master world state and communicates state updates to the clients. In the default state, without latency compensation, all player input is sent to the server, the server applies the input to the game world and sends the new world state to the client which renders the state for the player.

3.2 Testbed Setup

We setup the game for our user study in a dedicated, on-campus computer lab. The testbed setup is depicted in Figure 2. The server hosts the game and is connected via high-speed LAN to the client. The client and server are Alienware PCs with Intel i7-4790K CPUs @4 GHz with 16 GB RAM and an Intel HD 4600 graphics card. The client is equipped with a gaming mouse and monitor so as to minimize local system latency and maintain consistency. The client has a 25" Lenovo Legion monitor running at 1920x1080 pixels displayed at 16:9 and 240 Hz, with AMD FreeSync and a 1 ms response time. The mouse is a Logitech G502 12k DPI with a 1000 Hz polling rate. The clients and the server run Ubuntu 20.04 LTS, with Linux kernel version 5.4.

The local latency was measured with a 1000 frame/s camera (a Casio EX-ZR100) setup to capture the moment a user presses the mouse button and the resulting screen output. By manually examining the video frames, the frame time when the mouse is clicked is subtracted from the frame time the result is visible, giving the local system latency. This measurement method was done 10 times on our client, yielding an average base latency of 22 milliseconds, with a standard deviation of 5 milliseconds.

Local latency delays all input until resulting rendered output, whereas network latency delays receipt of the player's action at the server and subsequent server response to the client. Since the game server is authoritative, the client cannot update the position of an avatar until the server response has arrived. Thus, for the selection game without latency compensation techniques (as for all client-server games without latency compensation), local latency manifests similarly to network latency. Player orientation input until resulting avatar orientation change seen on the screen is delayed by at least the sum of the local latency and the network latency.

Our intent is to assess local latencies over ranges that might typically be found in personal computers, which range from about 25 milliseconds for a fast gaming system to around 100 milliseconds for a typical computer system [16]. We added latency to all mouse and keyboard input using EvLag [23] – an open-source tool for Linux that adds a constant amount of latency to any input device. Given our client has an average local latency of 22 milliseconds, EvLag adds either 3, 28, 53 or 78 milliseconds of latency for resulting total local latencies of 25, 50, 75, 100 milliseconds, respectively.

Similarly, our intent is to assess network latencies over ranges typically experienced by PC network game players, which can be near 0 milliseconds for a local area network (LAN) game, 100 milliseconds for a reasonable Internet connection, and 200 milliseconds for a slower Internet connection [37]. We added network latency to the server uplink and downlink equally using Linux tc with Netem[2] – a network control tool. The total round-trip network latency added to the client was either of 0, 50, 100 or 150 milliseconds.

Latency compensation techniques can mitigate the effects of network latency on game players. While there are many different types of latency compensation techniques, time warp and self-prediction are among the most commonly used in first-person shooter games [31]. To better understand and quantify how much each helps users in 3D selection tasks with network latency, we investigated four different latency compensation conditions: none, time warp only, self-prediction only, and both time warp and self-prediction. We implemented the different latency compensation techniques in our custom selection game. With self-prediction, the client predicts self movement and orientation. With time warp, the server resolves actions based on previous client game states when the action is triggered on the client. In commercial first-person shooter games played over the Internet, to avoid cheating, the server always makes decisions on game outcomes instead of the clients. Since our game is monitored by a proctor, cheating is not an issue, so for time warp, the client calculates the outcome of the frame based on the game state on the client. The client then notifies the server of the outcome. Upon receipt of the notification, the server updates the game state according to the outcome and synchronizes with the clients. In both our implementation and commercial implementations, timewarp effectively allows the player aim directly at the target – rather than "leading" a moving target by aiming in front of them as is required without timewarp – in order to hit the target. Either way, players need to wait a round-trip time from the client to the server to see the outcome displayed on the screen.

While the game is single player in that the player shoots a computer-controlled opponent, the lag compensation algorithms are experienced by the player as they would if they were playing against a human. The client-side prediction is made independently of the opponent action, and the time-warp state rollback on the server happens to match the player's game state. In other words, neither the client notification nor rollback state would change if there was a human controlled opponent. The actions of the opponent would likely change, but not the latency (and latency compensation) experienced by the player.

Target movement can change the game difficulty and affect player experience [35]. To better understand the effects of target movement, we studied three forms of motion for the enemy bot – normal that includes the movement described above (direction

[2]https://wiki.linuxfoundation.org/networking/netem

changes and jumping), normal but without jumping, and stationary without movement or jumping.

Table 1 summarizes the user study parameters.

Table 1: Experimental parameters for the user study.

Parameters	Values
Local latency	25, 50, 75, 100 (ms)
Network latency	0, 50, 100, 150 (ms)
Latency compensation	none, time warp, self-prediction, both
Opponent motion	stationary, normal without jump, normal

3.3 User Study Procedure

The study was approved by our University's Institute Review Board (IRB). Interested participants first filled out a screener questionnaire with questions on first-person shooter (FPS) game experience to help select participants with some prior familiarity with FPS games. Selected users were invited to the lab at a pre-set time. Users then signed a consent form and positioned themselves at the test computer.

Users first did a custom reaction-time test written in Javascript and launched via a Chrome Web browser. In the test, users waited for a screen color change then clicked the mouse as quickly as possible, doing this 10 times. The average of the 10 values provides a measure of reaction time.

Users started by playing a practice round without any added latency to get familiar with the game. This data was not analyzed. Users next played additional rounds, each with options for local latency, network latency, latency compensation techniques, and target motion, randomly shuffled. The conditions tested include:

A) Local latency: There are 3 conditions investigating local latency only (network latency of 0 ms, no latency compensation, and normal bot motion): local latencies of 50, 75 or 100 ms.

B) Network latency and latency compensation: There are 12 conditions investigating network latency with and without latency compensation (with local latency of 25 ms and normal bot motion): all combinations of network latencies of 50, 100 or 150 ms and four latency compensation conditions: none, time warp, self-prediction or both.

C) Local latency and network latency: There are 3 conditions assessing latency compensation with local latency and network latency (network latency of 100 ms, local latency of 100 ms and normal bot motion): time warp, self-prediction or both.

D) Target motion: There are 3 conditions investigating target motion (network latency and local latency of 100 ms, no latency compensation): normal bot motion, normal bot motion without jumping, or stationary bots.

Each condition above was repeated 3 times, for a total $(3 + 12 + 3 + 3) \times 3 = 63$ rounds plus the practice round. In addition to these 64 rounds, users play 5 baseline rounds with no latency, no latency compensation techniques and normal motion only. The 5 baseline rounds are uniformly distributed between all other rounds to ensure the consistency of player performance and test if there is fatigue giving the large number of game rounds. In total, each participant played $64 + 5 = 69$ rounds.

After each round, users provided a subjective Mean Opinion Score (MOS) rating on a discrete 5-point Likert scale about their experience: "Rate the quality of the previous game round". Players chose from 5 options: Excellent, Good, Fair, Poor or Bad. After completing the survey, the next round would commence when the user was ready, but users could take as much time as needed before starting the subsequent round.

It took each user about one hour to complete all the tasks in the study. A user study proctor was available for questions and trouble-shooting for the duration.

After completing all the game rounds, users were given a questionnaire with additional demographics questions about gamer experience – average time spent playing games and self-rated expertise with computer games.

Study participants were solicited via university email lists. All users were eligible for a raffle to win a $25 USD Amazon gift card upon completion of the study, and many users received academic credit for relevant classes in which they were enrolled.

4 ANALYSIS

This section first summarizes the demographics of our participants, then analyzes player performance and QoE for latency conditions without latency compensation, followed by analysis of latency compensation, and lastly, the effects of target motion on player performance.

4.1 Demographics

Thirty-nine (39) users were recruited and participated in total. This section provides summary demographics for the participants.

Table 2: Demographic information

Users	Age (yrs)	Gender	Gaming / week (hrs.)	Gamer Self-rating	FPS Self-rating	Reaction-time (ms)
39	20.0 (3.0)	♂31 ♀7 ○1	12.5	3.4 (1.3)	3.0 (1.2)	198.4 (16.9)

Table 2 summarizes the demographic information for the user study participants. Gamer and first-person shooter (FPS) self-rating are on a five-point scale, 1-low to 5-high. For age, gamer self-rating, FPS self-rating, and reaction times, the mean values are given with standard deviations in parentheses. Ages ranged from 18-32 years old but with the large majority of typical college age. Gender breakdown is predominantly male (31 males), but does reflect the gender breakdown of first-person shooter game players (only about 7% of first-person shooter gamers are women [20]) and our sample pool of students at our university. Half of the participants played 10 or more hours of computer games per week. User self-ratings in general computer games slightly skews towards above the midpoint (mean 3.4), with self-rating in FPS games slightly lower (mean 3.0). Most participants majored in Robotics Engineering, Computer Science, or Game Development. For the reaction time, the base local latency (22 ms) was subtracted from all reaction time trials and the

resulting reaction times averaged for each user. Reaction times are mostly fast (with an average of about 200 ms), typical of computer game players [9].

Table 3: Player skill, first hit time and QoE

Skill	FPS Self-rating	N	First hit time	QoE
Low	1,2	15	4.2 (2.4)	3.1 (1.1)
Medium	3	11	3.9 (3.5)	3.4 (1.1)
High	4, 5	13	3.7 (3.1)	3.3 (1.2)

Table 3 summarizes first hit elapsed time and QoE of players from different skill groups. Players with FPS self-rating at 1 and 2 are classified as low skill, players with FPS self-rating at 3 are classified as medium skill and players with FPS self-rating at 4 and 5 are classified as high skill as suggested by earlier work [27]. From this table, higher skill players tend to have shorter first hit time on average and medium and high skill players generally have higher QoE than low skill players.

4.2 Without Latency Compensation

We first analyze player performance without latency compensation. Analysis is for conditions where the opponent bot moves normally (test conditions A-C, but not D in Section 3). Since previous work [28] shows that local latency and network latency have an equivalent impact on players in the absence of latency compensation, we combine the conditions of local latency and network latency into seven different total latencies: 25, 50, 75, 100, 125, 175 and 200 ms. Thus, the analysis in this section pertains to selecting a moving, 3D target where all input – whether from the local computer system or from the network and remote servers – is delayed.

Figure 3a depicts the elapsed time required to select (hit) the target versus the total latency. The x-axis is the total latency (network plus local) in milliseconds and the y-axis is elapsed time in seconds. The circles are the mean elapsed times bounded by 95% confidence intervals and the dashed lines are linear regressions through the mean values. The blue points and line are for the first hit and the green points and line are for the second hit (the target takes 2 hits for a kill and the round to end). From the graph, the elapsed times increase with latency, indicating that latency makes it harder for players to shoot an opponent. The linear regressions fit the means well, with R^2 0.96 and $p < 0.001$ for the first hit (blue) and R^2 0.94 and $p < 0.001$ for the second hit (green). The blue line is above the green line meaning the first hit takes longer, on average, than does the second hit and the difference in slopes suggest the first hit is impacted slightly more by latency than the second hit. As a take-away, an increase in total latency by 100 ms increases elapsed time by about 2 seconds for the first hit, and 1.5 seconds for the second hit.

Figure 3b depicts accuracy versus total latency. The x-axes and trendlines are as for Figure 3a, but the data here for the y-axis is the accuracy (shots fired divided by shots taken) as a percent. From the graph, player accuracies for first and second hits degrade with latency, and the steeper slope of the blue line indicates that latency has higher impact on the first hit. The linear regressions fit the means well, with R^2 0.92 and $p = 0.001$ for the first hit (blue). R^2 0.87 and $p = 0.002$ for the second hit (green). As a take-away, an increase in total latency by 100 ms degrades accuracy percent by about 13% for the first hit, and about 8% for the second hit.

Note, the second hit times are affected by the weapon firing rate (i.e., the minimum time between successive shots – 250 ms in our study). Moreover, many first-person shooter games (although not ours) have weapon recoil that reduces accuracy and increases elapsed time for successive shots. While both factors – firing rate and recoil – are important for first-person shooter game performance for second and subsequent shots, they are not the focus of our current study. Hence, for all subsequent analysis, we analyze the first hit only.

Figure 3c shows an analysis of quality of experience (QoE) versus total latency. The QoE is from answers to the question "Rate the quality of the previous game round" from 1 (low) to 5 (high). The y-axis is the QoE and the x-axis is the total latency. Each point is the QoE averaged over all users, bounded by 95% confidence intervals. The line is a regression through the mean values with an R^2 of 0.62, $p = 0.064$. From the graph, latency degrades player experience, dropping about 0.4 points (on a 5-point scale) every 100 ms.

Player QoE and first hit elapsed time has a medium correlation with an R at -0.32.

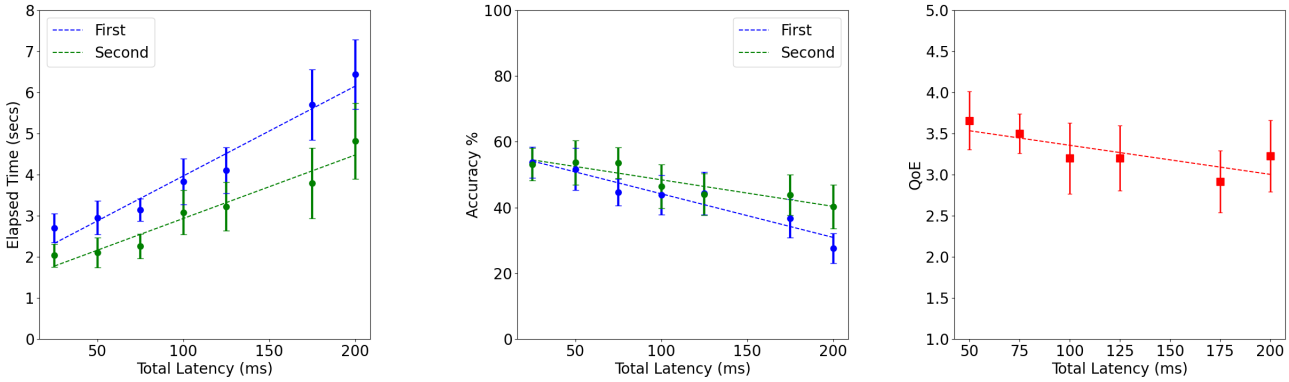
4.3 With Latency Compensation

Latency compensation techniques can mitigate the impact of network latency on players and are widely used in computer games, and self-prediction and time warp are commonly used latency compensation techniques in first-person shooter games [31]. We compare four different latency compensation condition: none, only self-prediction, only time warp, and both self-prediction and time warp. Local latency is kept at a minimum (25 ms), and network latency varies: 0 ms, 50 ms, 100 ms and 150 ms.

Self-prediction [18] estimates game state based on player input, but before getting confirmation from the server. In first-person shooter games, self-prediction primarily helps a player's movement in the presence of network latency where an avatar moves and changes direction in response to player input as if there is no network latency. In our first-person shooter game, player prediction provides immediate feedback to the player for changing orientation (aiming) without having to get confirmation from the server.

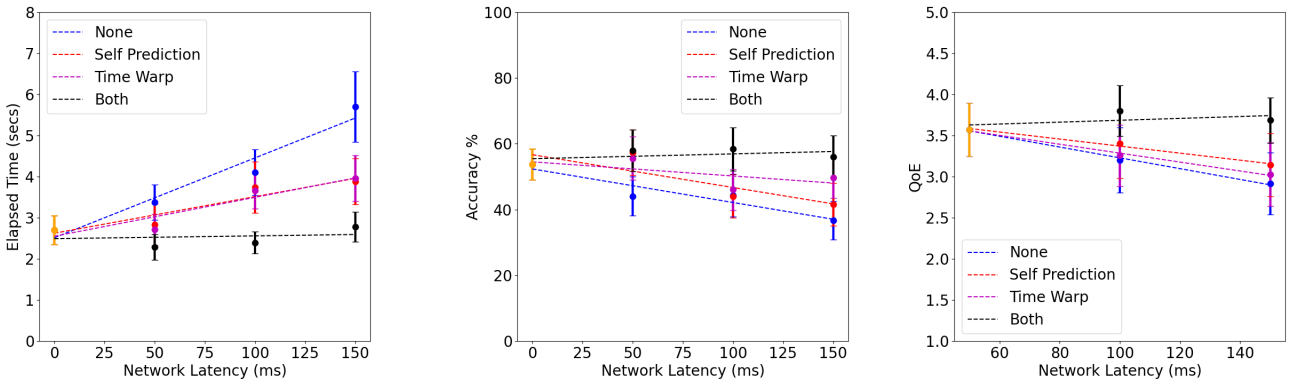
Time warp [43] rolls back game state on the server to when the player action occurred on the client, applies the action, then rolls the game state forward to the current time. In first-person shooter games, the server decides whether a player hits a target based on the previous game state when the player fired the shot, allowing players to aim directly at the target as if there is no network latency.

Figure 4a depicts elapsed time versus network latency where the axes and data are as for Figure 3a. The data is separated out (means and trendlines) by the four latency conditions: blue is without latency compensation, red is for self-prediction, purple is for time warp, and black is for both self-prediction and time warp. In the graph, the blue line has the steepest slope, showing that latency



(a) Elapsed time **(b) Accuracy** **(c) QoE**

Figure 3: Versus network latency without latency compensation



(a) Elapsed time **(b) Accuracy** **(c) QoE**

Figure 4: Versus network latency with latency compensation

has the most impact on elapsed time when there is no latency compensation. The red line and the purple lines have shallower slopes and are comparable, indicating that each technique individually has about the same ability to mitigate network latency. The black line is almost flat, indicating that both techniques together can nearly completely overcome the effects of network latency on elapsed time. The blue, red and purple lines fit their respective mean values well, with R^2 0.95, 0.89 and 0.88 and $p = 0.025$, 0.057, 0.061 respectively. The black line is almost flat with R^2 0.03 and $p = 0.82$, indicating that with both self-prediction and time warp, latency does not have statistically significant effect on elapsed time.

Figure 4b depicts the same data, but for accuracy. As for the elapsed time analysis, the uncompensated line (blue) is steeper than self-prediction (red) and time warp (purple), and with both self-prediction and time warp (black) the trend line is flat. The blue, red and purple line-fits have R^2 0.88, 0.32 and 0.49 and $p = 0.060$, 0.43, 0.30 respectively, while the black line has R^2 0.57 and $p = 0.24$ – with both self-prediction and time warp, network latency does not have significant impact on accuracy.

Figure 4c depicts QoE versus network latency with compensation techniques. The graph is the same as Figure 3c but the x-axis is only network latency and the data is separated by latency compensation condition. From the graph, as for player performance, QoE degrades the most with latency without compensation (blue), while self-prediction (red) and time warp (purple) both ameliorate the effects of network latency on QoE. The slightly steeper slope of the purple line compared to the red line indicates self-prediction helps QoE more than does time warp. The blue, red and purple lines have R^2 0.99, 0.99 and 0.99 and $p = 0.048$, 0.073, 0.058, respectively. The black line for QoE is almost flat with R^2 0.25 and $p = 0.67$ – with both time warp and self-prediction, latency has little impact on QoE.

As a take away, with both self-prediction and time warp together, latency does not appreciably affect player performance nor QoE for 3D target selection.

4.4 Local Latency and Network Latency

The latency compensation benefits from Section 4.3 can only mitigate the network latencies experienced in Section 4.3 but cannot help with the local latencies. We analyze how effective latency compensation is when there are also high local latencies (test condition C from our methodology).

For Figure 5 the network latency is fixed at 100 ms, while the local latency – either 25 ms or 100 ms – is on the y-axis.

From these graphs, the relative trends in player performance are the same – performance is worse without compensation and using both techniques helps more than each individually. However, even when using both latency compensation techniques together, performance with a local latency of 100 ms is still worse than with a local latency of 25 ms since the latency compensation techniques only mitigate the network latency and not the additional 75 ms of local latency.

In Figure 5a, without latency compensation, there is a significant difference between player elapsed time for 25 ms and 100 ms with $p < 0.001$. With only self-prediction or time warp, there is no significant different between player elapsed time at 25 ms and 100 ms with $p = 0.17$ and $p = 0.005$, respectively. With both compensation techniques on, there is significant difference between player elapsed time at 25 ms and 100 ms with $p < 0.001$. In Figure 5b, without latency compensation, there is significant difference between 25 ms and 100 ms with $p < 0.001$. With only self-prediction or time warp, there is no significant different between 25 ms and 100 ms with $p = 0.12$ and $p = 0.36$, respectively. With both compensation techniques, there is no significant difference between 25 ms and 100 ms with $p = 0.094$.

For QoE, the individual techniques cannot fully overcome the 100 ms of network latency so player QoE is about the same, although both techniques together improve QoE from about 3.25 to 3.75 when there is only 25 ms of local latency. In Figure 5c, without latency compensation, there is no significant difference between 25 ms and 100 ms with $p = 0.92$. With only self-prediction or time warp, there is no significant different between 25 ms and 100 ms with $p = 0.75$ and $p = 1.0$, respectively. With both compensation techniques, there is significant difference between 25 ms and 100 ms with $p = 0.041$.

4.5 Target Motion

While players in first-person shooter games usually shoot at moving opponents, some opponents deliberately jump to avoid being shot while others deliberately stand still, either unaware they are being shot at or to better aim their own weapons.

The data from test condition D in our methodology lets us assess how much these motion variants impact target selection. For this condition, only the motion varies – normal, normal without jumping, and stationary – whereas local latency and network latency are fixed at 100 ms, each.

Figure 6a depicts elapsed time versus the target motion condition. The x-axis is the motion condition and the y-axis is the elapsed time. The circles are mean values and the bars are 95% confidence intervals. Figure 6b is as for Figure 6a, but the y-axis is accuracy instead of elapsed time. From the graphs, player performance is significantly better – about 1/2 the elapsed time and twice the accuracy – when the target is still. Moreover, jumping – a commonly used tactic by first-person shooter opponents – does not significantly degrade the shooting performance of players shooting at the jumper. While we believe these results likely hold in other first-person shooter games, the degree to which they hold will depend upon the avatar speeds, and frequency and height of the jumping and frequency of direction changes.

4.6 Models

Analytic models can help generalize results beyond the necessarily narrow range of conditions tested in a user study. In our case, this means generalizing to latencies other than one of the 7 discrete values used in our experiments. Analytic models can also be used to help with game design, where predicting player performance with latency can be used to adjust the game parameters [32] and change in-game attributes [19, 42, 46] in order to accommodate latency. Moreover, analytic models may be useful for discrete event simulations [24], where game performance can be selected using the model and applied to a simulated game outcome. In our case, a model of 3D target selection with latency could be combined with models of navigation with latency [25] in order to simulate moving and shooting in a first-person shooter game. This may enable predictions of player performance over a broad range of latency conditions, as well as other in-game conditions such as specific game parameters (e.g., weapon attributes, movement speeds).

Since the intent of 3D target selection is to click on the target as fast as possible, we analyze and then model the elapsed time distributions with latency. For now, we only consider the rounds with normal target movement and without latency compensation.

Figure 7a depicts the cumulative distribution functions (CDFs) of elapsed time. The x-axis is length of elapsed time in seconds and the y-axis is the cumulative distribution. The data is grouped for three[3] different total latency conditions. From the graph, there is some visual separation of the lines based on latency, with lower latencies generally having shorter elapsed times (the lines are shifted up and to the left).

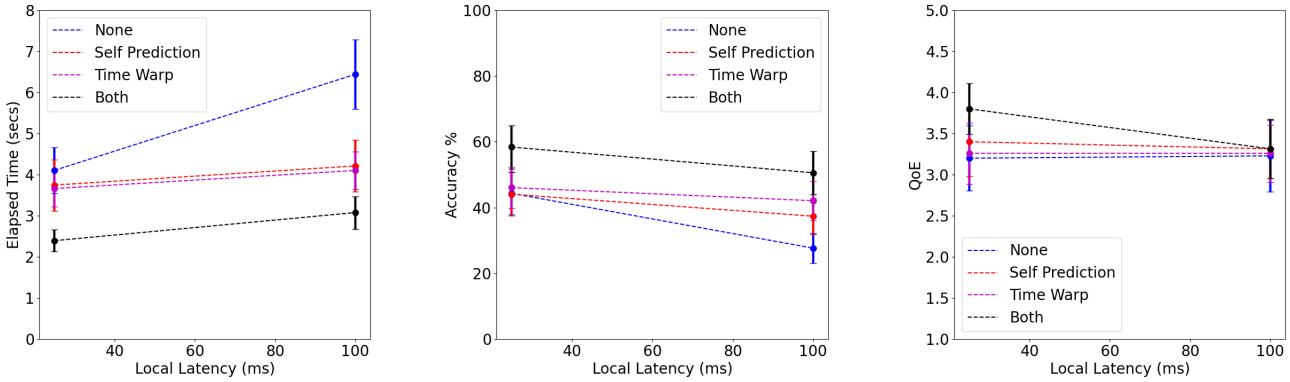
Figure 7b depicts the CDFs of elapsed time with latency compensation at 175 ms, as an example. In the graph, blue is for no compensation, red is for self-prediction, purple is for time warp, and black is for both techniques. Since there is no latency compensation in Figure 7a, the blue line in Figure 7b is the same as the green line in Figure 7a. From the graph, there is some visual separation of the lines based on latency compensation techniques, with the "both" condition having the shortest elapsed times, self-prediction only and time warp only having slightly longer elapsed times and "none" having the longest elapsed times.

Since the CDF distributions appear have an exponential shape, we fit an exponential function to the data. The CDF (p) of the exponential distribution is described by:

$$p = 1 - a * exp^{-(b \cdot L + c) * T} \quad (1)$$

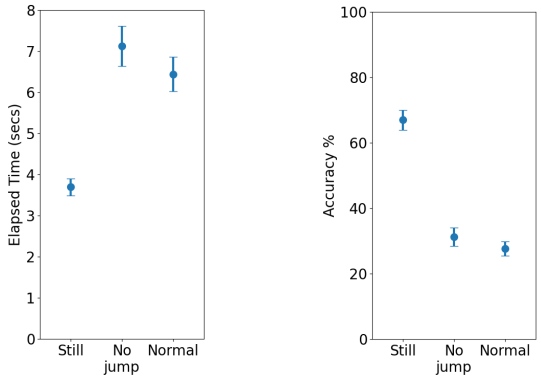
where L is the total latency (in milliseconds), T is the elapsed time (in seconds) and a, b and c are constants.

[3]The other latency conditions are not shown on the graph to make it readable.



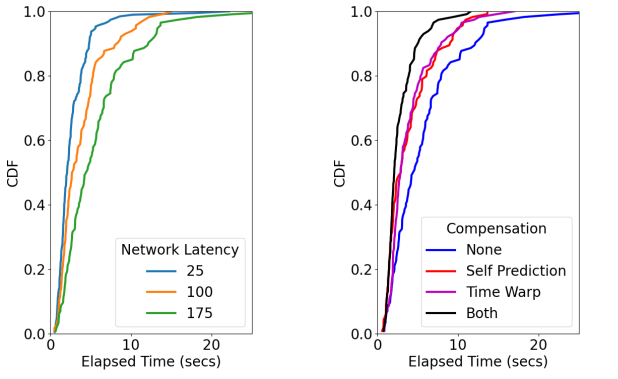
(a) Elapsed time (b) Accuracy (c) QoE

Figure 5: Versus local latency with 100 ms network latency



(a) Elapsed time (b) Accuracy

Figure 6: Versus target motion



(a) Without latency comp. (b) With latency comp. and 175 ms network latency

Figure 7: Distributions of first hits

The exponential functions fit the distribution data well, with an average R^2 0.98 and average root mean-square error (RMSE) of 0.041. For reference, the model parameters are provided by Table 4.

Table 4: Models for elapsed time distributions (Equation 1)

Latency compensation	a	b	c	R^2	RMSE
None	1.34	-1.57	0.51	0.99	0.033
Self-prediction	1.33	-0.97	0.50	0.98	0.039
Time warp	1.39	-1.16	0.53	0.98	0.042
Both	1.54	-0.52	0.60	0.97	0.049

5 DISCUSSION

The analysis of the first versus second shot matches expectations based on Fitts' Law [11]. The distance the player must move the reticle is farther for the first hit than the second, since for the second shot the mouse starts on the target and only has to move (track) until the weapon cooldown (250 ms) before firing again.

The performance of the player with latency compensation somewhat matches expectations, as well. Specifically, with self-prediction, the player's input is immediately acted upon by the client making the game responsive as if there is no additional network latency. With time warp, the player can aim directly at the opponent enabling shot resolution as if there is no additional network latency. Combined, the game feels as if there is no extra latency and this is reflected in the QoE scores. Corresponding to this, the player performs as well as they would without latency.

However, the fact that self-prediction or time warp by themselves provide less benefit is expected, but had not been measured. Basically, each technique provides about "half" the performance benefit of the combined pair, albeit neither provides much in the way of QoE benefit which is somewhat surprising. This lack of QoE benefit when used individually may be because even a little bit of latency has a detrimental effect on how the game feels, so compensation needs to overcome all latency for players to perceive the benefit.

As noted, FPS players often jump during a firefight to avoid being shot (i.e., presuming a randomly moving target is more difficult to aim at). However, based on our results, the selection times for hitting a moving and jumping target versus just hitting a moving target without jumping are similar, so this casts doubt on jumping as a dodging technique. In fact, jumping may make it harder to *aim*, an aspect we did not assess, and so players should consider jumping as an avoidance technique carefully before using in-game.

The analytic models provided, although simple, reveal that selection times are heavy-tailed in that average times can be relatively low – meaning hitting a target can happen quickly – but sometimes it takes a long time to sight and hit an opponent. And these tails get heavier with latency, which may inform players in their expectations and planning in play. On a related note, Fitts' Law [11] relates elapsed time to select a target with target distance and size. But in Fitts, the target size is fixed whereas our 2D size varies with the relative movement of the 3D target. Our results could provide a foundation for a revised law that relates elapsed time to 3D distance (and 3D size).

Considering moving and shooting are the main player actions in first-person shooter games, the analytic model of 3D target selection can be combined with 3D navigation model from our previous work [25] to simulate a wide range of first-person shooter game scenarios. Combat could be simulated as follows: moving to shoot and avoid being shot are done with successive, alternative "in-sight" and "out-of-sight" time windows generated with the navigation models. In each of the in-sight windows, the time to aim and shoot at the opponent is generated from the elapsed time models. If the elapsed time is contained within the in-sight window, the player hits the shot whereas if the elapsed time is longer than the end of the in-sight window, the player misses the shot. The simulations should be validated using actual first-person shooter game data. Once validated, the model-based simulation could be used for exploration of the first-person shooter games without the need for expensive and time-consuming user studies.

6 LIMITATIONS

Our methodology intentionally had users play against a bot. The movement of the target can alter the difficulty of the game and hence affect player performance and experience. Although the movement of the bot is simulated from real player data from the first-person shooter game (CS:GO), the results may not generalize to all FPS games which may differ in their target motion parameters.

In our custom game, the player only has a pistol as the weapon. However, play style and strategy can vary with type of weapon, which, in turn, may result in different elapsed times and accuracies. Similarly, weapon accuracies and firing rates different than the ones in our study may have alternate performance data.

As noted in Section 4.1, our sample is skewed towards young males. While this may reflect the gender and age breakdown present in some first-person shooter games today, the results reported may not be indicative of players outside of this demographic.

There might be learning effects where players become more familiar with the game and perform better at later rounds, regardless of the latency. While we did not explicitly observe learning effects in our five baseline values, our shuffled test conditions across all rounds should minimize any learning effects on specific conditions.

Serious game players often customize the software settings on their computers and games to suit their personal play preferences. However, since customizations would have created a difference in test conditions between users, we did not allow any changes to the computer settings. This holds for other game configurations, too, such as other mice, keyboards or monitors.

7 CONCLUSION

Understanding the effects of latency on first-person selection can help inform game design and development techniques to mitigate latency's effects, and also generalize results to a broad range of first-person shooter games through modeling and simulation. This paper presents results from a user study on first-person selection under controlled latency conditions. We isolated selection in first-person games via a custom 3D target selection game where players took a fixed position, then selected (shot at) avatars in a matter akin to first-person shooter games We also investigated common latency compensation techniques applied in first-person shooter games and studied different types of target movement. We setup our game in a private, local area network and control the local latency and network latency. Thirty-nine (39) participants played our custom game for 69 rounds across 22 different latency, latency compensation techniques and target motion conditions (a total of about 60 minutes of gameplay each), providing objective player performance data (elapsed time, hit/miss ratio) via log files and subjective opinion data. (Quality of Experience) via surveys.

Analysis of the results shows that across the range of total latencies studied, player performance and quality of experience both degrade linearly as latencies increase from 25 ms to 200 ms. Specifically, elapsed times to select at 25 ms of latency average about 60% shorter than elapsed times at 200 ms. Over this same range, Quality of Experience (QoE) decreases about 0.4 (on a 5-point cale) with every 100 ms of latency. Time warp and self-prediction both mitigate the effects of latency, and, when applied together, can eliminate the effects of latency on both player performance and QoE – i.e., players can feel and perform as if there is no network latency when time warp and self-prediction are both used.

Our future work is to investigate the impact of latency on first-person shooter games by simulating first-person shooter behavior using the models in this paper and additional models for first-person navigation [25]. Once developed, such simulations would need to be validated before being used to generalize performance over a wide range of first-person shooter game configurations. Other future work could explore how first-person selection performance varies by player skill, using either self-rated skill [27] or measures of proficiency in a selection task. Other future work can investigate first-person selection with different types of weapons, target motion and map features. Our methodology could be used for gaming actions in other games genres, e.g., Multiplayer Online Battle Arena (MOBA) games like *DOTA 2* (Valve, 2013) and *League of Legends* (Riot Games, 2009), and Real-Time Strategy (RTS) games like *Starcraft* (Blizzard, 1998). In such a case, individual game actions would need to be isolated for the game and then evaluated.

REFERENCES

[1] 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*. Springer-Verlag, Berlin, Heidelberg, 97–106.

[2] Grenville Armitage. 2003. An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3. In *Proceedings of the 11th IEEE International Conference on Networks (ICON)*. Sydney, Australia, 137–141.

[3] 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, OR, USA.

[4] Yahn W. Bernier. 2001. Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization. In *Proceedings of the Game Developers Conference (GDC)*. CMP Media, LLC, San Francisco, CA, USA, 13 pages.

[5] Mark Claypool. 2018. Game Input with Delay—Moving Target Selection with a Game Controller Thumbstick. *ACM Trans. Multimedia Comput. Commun. Appl.* 14, 3s, Article 57 (June 2018), 22 pages. https://doi.org/10.1145/3187288

[6] 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)*. Springer International Publishing, Reykjavik, Iceland, 226–237.

[7] 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.

[8] Matthias Dick, Oliver Wellnitz, and Lars Wolf. 2005. Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games. In *Proceedings of 4th ACM Workshop on Network and Systems Support for Games (NetGames)*. Hawthorn, NY, USA, 1–7.

[9] Matthew Dye, C. Shawn Green, and Daphne Bavelier. 2009. Increasing Speed of Processing with Action Video Games. *Current Directions in Psychological Science* 18, 6 (Dec. 2009), 321–326.

[10] E$ports Earning. 2019. Prize Money, Results, History, Statistics. Online: https://www.esportsearnings.com/. (Accessed January 5, 2021).

[11] 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.

[12] 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.

[13] 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, 1–9.

[14] Christina Gough. 2020. eSports Audience Size Worldwide from 2018 to 2023. Statista. Online: https://tinyurl.com/y3tffxzo. (Accessed September 17, 2020).

[15] Oliver Hohlfeld, Hannes Fiedler, Enric Pujol, and Dennis Guse. 2016. Insensitivity to Network Delay: Minecraft Gaming Experience of Casual Gamers. In *Proceedings of the International Teletraffic Congress (ITC)*. IEEE, Würzburg, Germany, 31–33.

[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 ACM Conference on Human Factors in Computing Systems (CHI)*. Seoul, Republic of Korea, 135–144.

[17] Benjamin F Janzen and Robert J Teather. 2014. Is 60 fps better than 30? The impact of frame rate and latency on moving target selection. In *Extended Abstracts on Human Factors in Computing Systems*. 1477–1482.

[18] 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 ACM International Conference on Interactive Surfaces and Spaces (ISS '17)*. Association for Computing Machinery, Brighton, United Kingdom, 230–239. https://doi.org/10.1145/3132272.3134138

[19] Injung Lee, Sunjun Kim, and Byungjoo Lee. 2019. Geometrically Compensating Effect of End-to-End Latency in Moving-Target Selection Games. In *the ACM Computer-Human Interaction Conference (CHI)*. Glasgow, Scotland, UK.

[20] Nick Lee. 2017. Beyond 50/50: Breaking Down The Percentage of Female Gamers by Genre. Online: https://quanticfoundry.com/2017/01/19/female-gamers-by-genre/. (Accessed September 5, 2021).

[21] Steven WK Lee and Rocky KC Chang. 2017. On 'Shot Around a Corner' in First-person Shooter Games. In *Proceedings of the IEEE International Workshop on Network and Systems Support for Games (NetGames)*. Taipei, Taiwan.

[22] Wai-Kiu Lee and Rocky KC Chang. 2015. Evaluation of lag-related configurations in first-person shooter games. In *Proceedings of the IEEE International Workshop on Network and Systems Support for Games (NetGames)*. Zagreb, Croatia.

[23] Shengmei Liu and Mark Claypool. 2021. EvLag - A Tool for Monitoring and Lagging Linux Input Devices. In *Proceedings of the ACM Multimedia Systems Conference (MMSys)*. Istanbul, Turkey.

[24] Shengmei Liu and Mark Claypool. 2021. Game Input with Delay - A Model of the Time Distribution for Selecting a Moving Target with a Mouse. In *Proceedings of the 27th International Conference on MultiMedia Modeling (MMM)*. Virtual Conference.

[25] Shengmei Liu and Mark Claypool. 2022. The Impact of Latency on Navigation in a First-Person Perspective Game. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*. New Orleans, LA, USA, 11 pages.

[26] Shengmei Liu, Mark Claypool, Andy Cockburn, Ragnhild Eg, Carl Gutwin, and Kjetil Raaen. 2021. Datasets: Moving Target Selection with Delay. In *Proceedings of the 12th ACM Multimedia Systems Conference*. 320–326.

[27] Shengmei Liu, Mark Claypool, Bhuvana Devigere, Atsuo Kuwahara, and Jamie Sherman. 2020. 'Git Gud!' - Evaluation of Self-Rated Player Skill Compared to Actual Player Performance. In *Proceedings of the ACM CHI PLAY*. Virtual Conference.

[28] Shengmei Liu, Atsuo Kuwahara, James Scovell, Jamie Sherman, and Mark Claypool. 2021. Comparing the Effects of Network Latency versus Local Latency on Competitive First Person Shooter Game Players. In *Proceedings of the ACM Esports and High Performance HCI Workshop (EHPHCI)*. Virtual Conference.

[29] Shengmei Liu, Atsuo Kuwahara, James Scovell, Jamie Sherman, and Mark Claypool. 2021. Lower is Better? The Effects of Local Latencies on Competitive First-Person Shooter Game Players. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*. Yokohama, Japan, 12 pages.

[30] Shengmei Liu, Atsuo Kuwahara, James Scovell, Jamie Sherman, and Mark Claypool. 2021. The Effects of Network Latency on Competitive First-Person Shooter Game Players. In *Proceedings of Quality of Multimedia Experience (QoMEX)*. Virtual Conference.

[31] Shengmei Liu, Xiaokun Xu, and Mark Claypool. 2022. A Survey and Taxonomy of Latency Compensation Techniques for Network Computer Games. *Comput. Surveys* 54, 11s (Feb. 2022). https://doi.org/10.1145/3519023

[32] Michael Long and Carl Gutwin. 2018. Characterizing and Modeling the Effects of Local Latency on Game Performance and Experience. In *Proceedings ACM CHI Play*. New York, NY, USA, 285–297.

[33] Michael Long and Carl Gutwin. 2019. Effects of Local Latency on Game Pointing Devices and Game Pointing Tasks. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*. Glasgow Scotland, UK, 1–12.

[34] I. Scott MacKenzie and Colin Ware. 1993. Lag as a Determinant of Human Performance in Interactive Systems. In *Proceedings the Conference on Human Factors in Computing Systems* (Amsterdam, The Netherlands). IOS Press.

[35] Andy Cockburn Mark Claypool and Carl Gutwin. 2020. The Impact of Motion and Delay on Selecting Game Targets with a Mouse. *ACM Transactions on Multimedia, Computing, Communication and Applications (TOMM)* 16, 2 (May 2020).

[36] Emmie Martin. 2018. Super Bowl Champs Will Win Thousands - but They'd Earn 130 Percent More If They Played Baseball. CNBC - Money Online: https://www.cnbc.com/2018/02/02/how-much-super-bowl-winners-get-paid-compared-to-world-series.html. (Accessed January 5, 2021).

[37] optimum.com. 2020. What Is Latency? Online: https://www.optimum.com/internet/what-latency. (Accessed Nov 15, 2021).

[38] Andriy Pavlovych and Carl Gutwin. 2012. Assessing Target Acquisition and Tracking Performance for Complex Moving Targets in the Presence of Latency and Jitter. In *Proceedings of Graphics Interface 2012* (Toronto, Ontario, Canada) *(GI '12)*. Canadian Information Processing Society, CAN, 109–116.

[39] Andriy Pavlovych and Wolfgang Stuerzlinger. 2011. Target Following Performance in the Presence of Latency, Jitter, and Signal Dropouts. In *Proceedings of Graphics Interface 2011* (St. John's, Newfoundland, Canada) *(GI '11)*. Canadian Human-Computer Communications Society, Waterloo, CAN, 33–40.

[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 Workshop on Network and Systems Support for Games (NetGames)*. Portland, OR, USA, 152–156.

[41] Kjetil Raaen and Ragnhild Eg. 2015. Instantaneous Human-Computer Interactions: Button Causes and Screen Effects. In *Proceedings of the 17th HCI International Conference*. Springer International Publishing, Los Angeles, CA, USA, 492–502.

[42] Saeed Shafiee Sabet, 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 *IEEE International Symposium on Multimedia (ISM)*. IEEE press, Taichung, Taiwan, 114–121.

[43] Cheryl Savery and T. C. Graham. 2013. Timelines: Simplifying the Programming of Lag Compensation for the Next Generation of Networked Games. *Multimedia Systems* 19, 3 (June 2013), 271–287. https://doi.org/10.1007/s00530-012-0271-3

[44] Robert J. Teather, Andriy Pavlovych, Wolfgang Stuerzlinger, and I. Scott MacKenzie. 2009. Effects of Tracking Technology, Latency, and Spatial Jitter on Object Movement. In *Proceedings of the IEEE 3D User Interfaces*. Lafayette, LA, USA.

[45] wepc.com. 2021. Video Game Industry Statistics, Trends and Data In 2021. Online: https://www.wepc.com/news/video-game-statistics/. (Accessed August 12, 2021).

[46] Xiaokun Xu, Michael Bosik, Adam Desveaux, Alejandra Garza, Alex Hunt, Cameron Person, James Plante, Joseph Swetz, Nina Taurich, Brian Clark, Doris Hung, Philip Lamoureux, and Mark Claypool. 2022. Compensating for Latency in Cloud-based Game Streaming using Attribute Scaling. In *Proceedings of Quality of Multimedia Experience (QoMEX)*. Lippstadt, Germany.