Quantifying and Mitigating the Negative Effects of Local Latencies on Aiming in 3D Shooter Games

Zenja Ivkovic, Ian Stavness, Carl Gutwin, and Steve Sutcliffe
Department of Computer Science, University of Saskatchewan
110 Science Place, Saskatoon, Canada, S7N 5C9
[zenja.ivkovic, ian.stavness, carl.gutwin, steve.sutcliffe]@usask.ca

ABSTRACT

Real-time games such as first-person shooters (FPS) are sensitive to even small amounts of lag. The effects of network latency have been studied, but less is known about *local latency*, the lag caused by input devices and displays. While local latency is important to gamers, we do not know how it affects aiming performance and whether we can reduce its negative effects. To explore these issues, we tested local latency in a variety of real-world gaming scenarios and carried out a controlled study focusing on targeting and tracking activities in an FPS game with varying degrees of local latency. In addition, we tested the ability of a lag compensation technique (based on aim assistance) to mitigate the negative effects. Our study found local latencies in the real-world range from 23 to 243 ms which cause significant and substantial degradation in performance (even for latencies as low as 41 ms). The study also showed that our compensation technique worked extremely well, reducing the problems caused by lag in the case of targeting, and removing the problem altogether in the case of tracking. Our work shows that local latency is a real and substantial problem – but games can mitigate the problem with appropriate compensation methods.

Author Keywords

Local latency; Lag; Aiming; Targeting; FPS games.

INTRODUCTION

Real-time competitive games require quick and timely responses to other players' actions and game events [24]. Any delays in the responsiveness of a game are disruptive because they create mismatches in timing between a player's actions and visual feedback of those actions, opponents' actions, and other game events. Latency on the game output forces players to react based on old information, and input latency adds to the time taken for actions to be registered. In both situations, player performance suffers, either because of a missed opportunity, incorrect response to an opponent's action, or reduced player control. As a further drawback, local latency may be imperceptible to the player, but still cause reduced performance and enjoyment [1].

CHI 2015, April 18 - 23, 2015, Seoul, Republic of Korea
Copyright is held by the owner/authors. Publication rights licensed to ACM.
ACM 978-1-4503-3145-6/15/04…$15.00
http://dx.doi.org/10.1145/2702123.2702432

Latency (or *lag*) arising from network communication has been studied in a number of contexts, and delays above 100 ms have been shown to be disruptive to racing games [19] and first-person shooter (FPS) games [2]. However, much less work has examined *local latencies* arising from input devices, displays, and software processing. Whereas the magnitude of network lag is generally decreasing, local latency is often increasing due to the use of wireless input devices and high-latency displays such as living room televisions. Our measurements suggest that local latency in these situations is often above 100 ms.

Previous studies have examined local latency in 2D pointing tasks (e.g., [16, 21]), but it should not be assumed that these results directly translate to FPS-style aiming tasks. FPS games are more complex both due to gameplay elements and the mechanism of aiming, which involves panning the view of the 3D world while the reticule remains centered in the screen (as opposed to translating a cursor over a 2D background). This mode of aiming requires users to process substantially more visual information, which tends to slow down feedback responses [7]. Looser et al. [15] reported that target acquisition in 3D FPS environments is 25% slower than in 2D scenarios. This decrease in performance suggests the effects of latency in 2D and FPS environments may also differ.

While some game genres, such as fighting games, force serious players to become aware of local latency, players of other genres are relatively less informed. Many players of FPS games know that local latency affects performance but the lack of quantification leads to misunderstanding and an underestimation of its impact. Without quantification, it is difficult to judge how severely lag affects different game tasks, to develop effective compensation techniques, to motivate consumers to consider latency when making game system purchases, and to motivate hardware and software developers to minimize local latency in their products.

To provide quantitative information about local latency we carried out three steps: we found the magnitude of local latency in typical real-world systems; we quantified the effects of latency on core FPS aiming tasks (acquisition and tracking); and we evaluated the effectiveness of aim compensation techniques for mitigating the effects of lag.

Our studies provide the following three findings:

- A sample of real-world systems (from PCs with LCD monitors, to gaming consoles with televisions) range in local latency from 23 to 243 ms.
- Local latency had a significant negative effect on both acquisition and tracking at latencies as low as 41 ms.
- Compensation techniques eliminated the negative effects of local latency in target tracking, and significantly reduced the effects in target acquisition.

We provide the first empirical evidence that local latency is a real and substantial problem for FPS games – even at very low levels, and levels lower than were measured in many real-world systems. In addition, we demonstrate compensation techniques that can reduce or even remove the effects of local latency. Our work shows that designers and publishers of FPS games (and other high-speed interactive systems) can benefit from compensating for local latency.

RELATED WORK

Latency and Local Latency

Latency is the measure of time delay between the sending and receiving of a message [8]. Latency is pervasive in digital systems and can vary greatly depending on the device (e.g., some LCD televisions have high latency due to built-in image processing). The negative effects of latency have been reported for a number of computer applications, such collaborative groupware [24], video scrubbing [18], digital sketching [3] and video games [20, 2].

Game system latency arises from a number of sources broadly classified as either network lag or local latency. Network latency occurs in multi-player video games due to delays in the network connection between players [22]. Network latency in video games has been well studied [19, 3]; however, effects of delay in a distributed system are different than the phenomena of local latency.

Local latency occurs in both multi- and single-player video games, because it is caused by delays in components of the game system that process user input and update the video display, rather than the network. Local latency causes delayed visual feedback of a player's own actions.

Local latency has recently become a subject of interest with new types of gaming systems that use high-latency devices, such as gesture input or televisions. Many games also rely on touch devices, such as tablets, mobile phones, or Nintendo's Wii U, that suffer from significant local latency. In a Fitts' law study with direct touch, Jota et al. found that latencies as low as 10ms reduced drag-and-point performance [11].

Aiming with Latency in 2D

Aiming movement can be decomposed into two phases: a *ballistic phase,* an open-loop action meant to arrive at the vicinity of the object; and a *controlled phase*, a closed-loop action relying on visual and kinetic feedback to carefully acquire the target [5]. In FPS games, two common aiming movements are target acquisition and target tracking.

Target acquisition uses both ballistic and controlled phases of movement. MacKenzie and Ware [16] measured acquisition time in four lag conditions using the Fitts' law paradigm. They found that lag had a significant negative effect on acquisition (up to 64% reduction in performance at 225 ms).

Target tracking involves keeping a pointing device over a moving target. Pavlovych et al. [21] examined the effect of lag on target tracking in 2D and found significant interaction between lag and longitudinal tracking errors. However, the 2D target motion in that study used Lissajous curves, which are smooth curves that do not match the horizontal ground-plane movements seen in FPS games. The curved target motion using Lissajous curves may aid the participant's ability to predict the target motion in high latency conditions and may not apply to FPS games.

Aiming in 3D

Target acquisition in full 3D VR environments with tri-variate targets has been shown to be negatively affected by latency [26]. Another 3D VR study suggested that, in the presence of latency, index of difficulty (ID) should include separate factors for target distance and target width [23].

Most 3D environments, such as FPS games, are 2D projections of a 3D scene, and therefore can be modeled as a bivariate pointing task [1,15], though pan-based pointing in 3D also requires angular measures in Fitts' law [12]. Although FPS pan-based targeting has been shown to obey Fitts' Law [15], no studies have been done to examine the effects of latency on targeting in 3D FPS environments.

FPS environments have unique properties that may make latency particularly disruptive. First, shooter games involve fast-paced game events, meaning that quick reflexes and precise aiming are paramount to performance. Second, the first-person perspective means that a player's movements cause the entire view of the 3D scene to shift behind a fixed cursor in the center of the screen. This causes much stronger visual changes (e.g., optical flow and motion parallax) as compared to 2D environments where a small cursor is moved across a fixed background.

Sources of Local Latency

A variety of sources can combine to result in high local system latency. For input devices, input that occurs between samples is not registered until the next time the input device is polled. For instance, USB mice by default are polled at a rate of 125 Hz (8ms intervals) in Windows XP and on, although gaming mice may set rates of up to 1000 Hz.

The display is a main contributor to local latency. Although specialized gaming monitors have low lag, most monitors average about 30-60 ms of latency. Televisions, which are popular in the game console community, often have significantly greater latency than monitors due to internal processing such as motion smoothing. The pixel response time in displays – that is, the time for a pixel to fully transition from one color to another, also contributes to the latency (5-10 ms). Additionally, displays almost always operate at 60

Hz refresh rates, which means that full screen updates cannot be shown any quicker than the refresh rate allows.

The game itself can cause significant amounts of latency. Input and output is processed in discrete intervals called frames, with frame rates typically ranging from 60 Hz in ideal cases (i.e., 16.7 ms periods), to 30 Hz or less, which is very commonly encountered on game consoles and slower computers. Any input received after the start of a frame period must wait until the next period, resulting in additional latency. Furthermore, games may perform input or output buffering, which causes more latency to be added and can contribute as much as 67ms or more.

Two additional software factors that affect latency at the output level are vertical sync (v-sync) and render-ahead. V-sync is used to synchronize game updates to display refreshes, which eliminates screen tearing, but it also caps the game frame rate to the display refresh rate. Without v-sync, a new frame can be rendered and sent to output while the monitor is the middle of a scan-out, which means that partial feedback can happen even quicker than the display refresh rate would suggest. Additionally, vertical sync locks the frame rate to factors of the refresh rate (e.g., a game capable of 55 fps on a 60Hz display would run at 30 fps). Triple buffering eliminates these discrete steps, but is only supported by OpenGL and not Direct3D (which is currently dominant). Instead, Direct3D uses render-ahead, which keeps a buffer of several frames to smooth output and eliminate tearing. The default buffer is three frames, which adds substantial latency [28].

Lag Compensation

Many multi-player games incorporate methods to reduce or compensate for *network* latency. Everquest 2 renders movement locally to reduce transmission rate and allow local queueing of events [6]. Quake IV uses prediction algorithms to compensate for lost data [27], a technique that is common for other FPS games. Another technique to improve aiming performance in lagged settings shows players their delay (a technique called Echoing [4]).

Aiming assistance is common in 3D games, but it is typically intended to reduce game difficulty or overcome controller difficulties, rather than to reduce the negative effects of local latency. The most common aim assists in games are ‘gravity’ techniques or ‘sticky targets’, where the cursor is attracted to the target when the player is in close proximity [25]. Others have suggested that eye-tracking technology could be used for aim assistance in games [9].

STUDY 1: LATENCY IN REAL-WORLD SYSTEMS

Although gamers share evidence when discussing game system latency [13], and significant marketing is employed to promote “low latency” input devices and displays, little empirical evidence exists for the local latency associated with different gaming setups. Minimizing latency in VR headset displays has been a recent focus of attention for new products, such as the Oculus Rift (www.oculusvr.com) that include specialized devices to measure latency from “motion-to-photon” while the device is in use. However, efforts to apply such measurement methodologies to typical FPS gaming systems have been uncommon.

We built our own latency measurement apparatus in order to sample a range of typical gaming setups and measure the amount of latency faced by gamers in real-world situations. We also used it with our experimental setup to measure the amount of latency in each latency condition of our user study. Our system measures the latency between an input mouse movement and resulting visible screen motion.

Latency Measurement Apparatus

The measurement apparatus used a high speed video camera (Canon ELPH 300HS digital camera, recording video at 240 frames per second) mounted on a tripod to capture the motion of the input device, e.g. the movement of a mouse on a flat wooden board elevated to the height of the display. This allowed both the mouse and the display to be captured in the camera frame at once.

To obtain a measurement, we ran the game system with a highly visible textured scene to allow for easy observation of movement on the screen. With the camera recording, we struck the input device sharply with a hammer to elicit a change in view direction. We measured the amount of latency by reviewing the recorded footage on a frame-by-frame basis, counting the number of frames elapsed between the mouse initially starting to move and the display changing. An LED light mounted on the device showed exactly when the movement started.

The reported latency period ended when any updates were first seen on the screen. Note that this could mean that only a small portion of the screen has updated at that point. For our experimental setup the display updated at 120 Hz, therefore it would take up to an additional 4 ms for half the screen to update and 8 ms for the screen to fully update.

Our camera makes discrete measurements with a precision of 4.2 ms and the display latency can have some jitter. For these reasons we report the average latency from 4 measurements for each of the real-world systems.

Local Latency of Typical FPS Gaming Systems

We used our latency measurement apparatus to determine the latency of a number of real-world systems. We selected systems from a variety of different types including gaming and office PCs and gaming consoles, and we used different types of input and output devices. Measurements were performed on a number of different games, using the same system settings that would be used during normal gameplay without modification. We found a wide range of latencies from 23ms on a PC with an ultra-fast gaming monitor, up to a surprising 243ms for the best-selling game GTA5, even on a relatively low-latency TV. Although our measurements are not intended to be exhaustive, they are a representative sample of real-world latency. Table 1 shows mean and standard deviation latency results in milliseconds.

System	Game	Input	Display	Lag (s.d.)
Win 7 (G)	Day Z	M, W	Asus VH242H (T)	117 (7)
OS X 10.9 (S)	Zombie count	M, WL	Toshiba PA3769 (VGA) (T)	158 (12)
Win 8.1 (G)	CS: GO	GM, W	Dell UP2414Q (I)	65 (4)
ArchLinux (G)	Path of Exile	GM, W	Overlord X270OC 100Hz (I)	32 (3)
Win 8 (G)	Quake 3	GM, WL	BenQ XL2420T 120Hz (T)	37 (2)
Win 8 (G)	Quake 3	GM, W	BenQ XL2420T 120Hz (T)	23 (2)
Win 8 (G)	Quake 3	GM, W	LG L226WTX (T)	64 (4)
Win 7 (S)	Cut the rope	M, W	Dell S2340Tt (I)	117 (6)
Wii U (C)	Windwaker HD	C, WL	Sony KDL55HX850 (V)	155 (16)
Wii U (C)	CoD Black Ops 2	C, WL	Sony KDL55HX850 (V)	130 (8)
PS4 (C)	Killzone SF	C, WL	Sony KDL55HX850 (V)	148 (8)
PS4 (C)	Watch Dogs	C, WL	Sony KDL55HX850 (V)	175 (16)
Xbox 360 (C)	Battlefield 3	C, WL	Sams. UN60EH6003F (V)	155 (23)
Xbox 360 (C)	GTA 5	C, WL	Sams. UN60EH6003F (V)	243 (60)
Wii U (C)	Bioshock Infinite	C, WL	Sams. UN60EH6003F (V)	192 (9)
Wii U (C)	Windwaker HD	C, WL	Sams. UN60EH6003F (V)	138 (17)

Systems: G = gaming computer (recent hardware, discrete GPU); S = standard computer; C = gaming console. **Input**: M = standard mouse; GM = gaming mouse; C = standard gamepad; W = wired, WL = wireless. **Displays**: T = twisted-nematic; I = in-plane switching or equivalent; V = LCD televisions (IPS with game mode on).

Table 1: Results of field latency sample.

STUDY 2&3: EFFECTS OF LOCAL LATENCY AND LAG COMPENSATION ON FPS TARGETING AND TRACKING

We designed controlled studies of two different FPS game tasks: target acquisition and target tracking. Our goals in these studies were to assess the effect of local latency on task performance, and to determine whether compensation techniques can limit the negative effects of lag.

Experimental Tasks

Our experimental system implemented 3D target tracking and target acquisition. These tasks are representative of primary activities in many 3D game environments: in FPS games, target acquisition is carried out when first encountering an enemy, and then a combination of acquisition and tracking is used until the enemy is eliminated.

Target Acquisition

In each trial, a roughly-spherical stationary target appears in the player‘s view. Players are instructed to aim at the target and click the left mouse button. If the target is hit, the player's view resets to the base position, and after a 500 ms delay, another target appears in a different position. The view is fixed during the 500 ms delay, ensuring each trial starts from the same position. If the target is missed, the trial continues until the target is hit, although the trial is repeated until completed without misses.

Each lag condition consists of 36 possible target positions within a virtual cone extending from the center of the base view position (Figure 1). The position order is randomized, but the same order occurs within every condition.

The 36 target positions were generated by different combinations of cone variables: angle (0°, 45°, 135°, 180°, 225°, 315° from horizontal), radius (74, 148, and 224 units), and depth from camera (224 units and 448 units, which effectively controls the size of the target). These variables are aggregated into six *Indices of Difficulty* (*ID*). Distance and radius are roughly equivalent to the width and distance variables in Fitts' Law [1]. Since rotation is used to aim, the units used are angular distances [15].

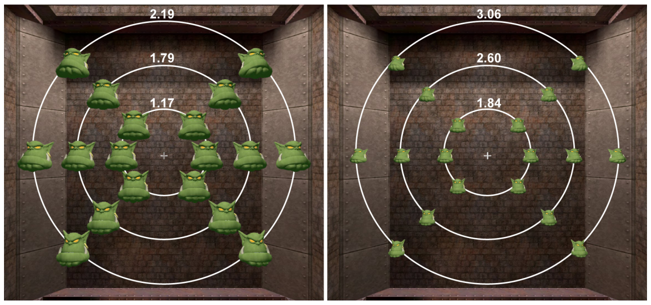

Figure 1: Target acquisition task showing all possible target locations and sizes. The six IDs (numeric labels) were chosen to be typical of targets in FPS games.

Target Tracking

In each tracking trial, a target appears at the same initial position in front of the player. To begin a trial, the player clicks on the target. The target moves horizontally (Figure 2) for 5 seconds and then the trial ends and the target resets. The player's goal is to keep aim on the target for as long as possible. The target turns green when the crosshair is over it. Although horizontal movement is a special case for mouse movement [14], it is the most commonly used tracking axis in FPS games since players often make side-to-side evasive movements along a ground plane.

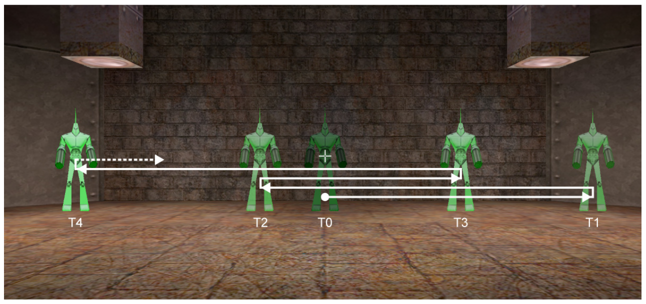

Figure 2: Target tracking task showing target movement from initial position (T0) through four direction changes (T1-4).

The only factor within a lag condition is target speed. Each sequence of trials begins with 4 slower trials at 240 units per second (u/s), followed by 4 trials at 320 u/s. The target's bounding box is 30 units wide and deep, with a height of 56 units. Direction changes take 140 ms to complete and occur at random intervals (taken uniformly from a period of 0.3 to 0.75 seconds, which is similar to player behavior in similar settings) with an average count of 10 reversals per trial. Target size and speed were chosen based on observations of existing games: the higher speed is based on player movement speeds in Quake III Arena, and the lower speed is based on more realistic slower-paced FPS games.

Questionnaire

Participants filled out a questionnaire with a seven-point Likert scale between each change of lag. Questions were "I performed well this round," "lag affected my performance," "this round was frustrating," and "the controls were laggy."

Artificial Local Latency

In order to control the amount of lag in our experiment setup, a system with low baseline lag was used with a technique to artificially increase the amount of latency in a controlled manner. Rather than varying each possible source of local latency, all latency was added at the output stage in order to reduce complexity. This approach is reasonable because input is rarely the primary source of local latency.

Five levels of system latency were used in the study, creating 11 ms, 41 ms, 74 ms, 114 ms, and 164 ms of lag. This covers a reasonable range of latency that matches well with levels found on PC systems in study one while retaining sufficient granularity to evaluate low levels of latency. The lowest achievable level (11 ms) serves as a baseline.

The lowest two latency conditions, 11 ms and 41 ms, were achieved using the natural latency of our system with v-sync disabled (11 ms) and enabled (41 ms). The higher three levels of latency were achieved with artificial latency, added to the graphical output by rendering frames sequentially into a ring buffer of textures. The choice of which frame to use from this buffer determines the amount of latency. To achieve baseline latency, frames are displayed immediately; to achieve two frames of latency, the texture in the buffer from two frames in the past is displayed.

Compensating for Local Latency

We developed compensation techniques based on existing aim assistance methods to mitigate the effects of local latency. The compensation involved aiming assistance that was proportional to the average expected latency. We devised our assistance so that it mitigates local latency rather than serving as a general form of performance boosting. In order to create targeted compensation schemes, we identified two primary ways that local latency affects the two tasks.

Target overshoot during target acquisition. During target acquisition, players tend to keep moving the crosshair toward the target until they observe that they have reached the target. If lag is present, people can overshoot the target because the visual state is behind the true game state. *'Sticky targets'* was chosen to compensate for lag in targeting because it directly combats the target overshoot effect by increasing the width of the target in motor space. Sticky targets reduces the mouse gain (also called *C:D* ratio [17]) while the aim is on the target, and is designed to decrease overshoot due to latency.

Direction changes during target tracking. Each time a target changes direction, players have to change their aiming movement. Perceiving direction changes under local latency can take substantially longer; and once players do react and attempt to change their aim, the target has moved even farther in the other direction. Another aspect to tracking is predicting future target positions and when it may change direction, if a pattern is apparent. Latency interferes with this process by delaying feedback regarding incorrect predictions. *Aim dragging*, an assist found in the Call of Duty games was chosen to compensate for lag in tracking because it provides assistance during direction changes, reducing the amount of time spent off target. *Aim dragging* causes the aim to be partially dragged in the direction of target movement. As in Call of Duty, aim dragging is active even if aiming near the target but not quite on it. The decreased-gain aspect of *sticky targets* is also applied, which mostly compensates for the fact that the tracking assistance could otherwise move the aim too quickly once added to the player's own input, and also provides overshoot protection.

Based on our previous experience with latency, we determined a general desired shape for the assistance strength curves. We then performed an initial pilot study with separate participants in order to tune the constant scalar and exponent terms applied to the lag variable. The formulas used for compensation are as follows:

$$\text{Sticky targets:} \qquad s = \frac{1}{1+10l^{1.2}}$$

where *s* is the change in mouse gain (with 1 being no change in gain) and *l* is the amount of lag in seconds. The effect is a roughly linear decrease in sensitivity up to about 100 ms, after which the effect decreases gradually to prevent the gain from being too low at high amounts of lag.

$$\text{Aim dragging:} \qquad d = 1 - \frac{1}{1+2.75l^{1.3}}$$

where *l* is the amount of lag in seconds and *d* is the strength of the drag, with 0 corresponding to no assistance and 1 being perfect assistance (auto-tracking). Again, the effect is approximately linear at first, with a gradual decreased slope at high latency. Table 2 shows the resulting assistance values at the latencies used in our experiment.

Factor	41 ms	74 ms	114 ms	164 ms
s	0.82	0.69	0.57	0.47
d	0.10	0.17	0.24	0.31

Table 2: Assistance strength at the experimental lag levels.

Experimental Conditions Used in the Study

Three factors were the same for acquisition and tracking tasks: *Lag*, *Block,* and *Compensation*. The tracking task also had a *Speed* factor for target speed, and the acquisition task had an *ID* factor for index of difficulty.

Lag. Five levels of latency were used in the study, corresponding to 11 ms, 41 ms, 74 ms, 114 ms, and 164 ms of lag. These numbers represent the average latency at that level, and although some frame to frame variation was present, it was minimal. The variation never exceeded one frame, and was on average much less than one frame.

Block. A sequence is a group of trials where all trials have the same level of lag (36 trials for acquisition, 8 for tracking). A block groups five sequences of sequentially increasing or decreasing lag. There were three blocks within each task for each compensated and non-compensated session. The first block increased lag from minimum to maximum. The second block then went from max to min, and the last block once again went from min to max. Blocks were used as a compromise to achieve a degree of counterbalancing, so that lag levels were spread throughout the experiment. This approach minimizes changes in lag at each adjustment, with the goal being to quantify the effect of lag once people have had time to get used to it, rather than observing rate of adaptation.

Compensation. This factor controlled whether lag compensation aiming assists were in use. The assistance strength was directly tied to the amount of lag in the current condition. At the minimum lag level, no assists were given.

Speed. Used only in the tracking task, this factor represented the speed of the target (240 units/s and 320 units/s).

ID. For target acquisition, six IDs were tested, consisting of combinations of target distance and cone radius.

Participants and Apparatus

Eighteen participants were recruited from a local university (15 male, 3 female). All participants had at least some experience with FPS games using a mouse. The same participants performed both studies over two sessions on consecutive days, with one session making use of lag compensation and the other being the baseline, non-compensated condition. The order of conditions was counterbalanced.

Half the participants started with the acquisition task, with the other half starting with tracking. Each task started with two half-duration blocks of training period (not included in the aforementioned 3 blocks).

Both studies were performed on a PC (Intel Core i7 3.2GHz CPU with NVidia GeForce 460GTX video) running GNU/Linux. In order to achieve the lowest minimum latency, we used a gaming mouse with a 1000Hz sampling rate (Razer DeathAdder 3.5G) and an ultra-fast gaming display with a 120Hz refresh rate (BenQ XL2420T).

The experiment was implemented as a C++ program utilizing the open-source Ogre3D (www.ogre3d.com) graphics engine using OpenGL as the rendering back-end. The software ran at a consistent 120 fps while v-sync was enabled.

Latency Measurement

We used the same apparatus as in Study 1 to measure lag in each experimental condition. To assess the latency jitter and ensure that our measurements were accurate, we made 18 measurements at each of the lowest latency levels and report the mean and standard deviation. We also verified that the higher latency conditions with artificial lag have the expected amount of additional latency. The mean baseline latency (no v-sync) was 11.4 ms (s.d. 2.28), and the first latency condition (v-sync on) was 40.8 ms (s.d. 3.13).

Design and Hypotheses

The acquisition study used a 2×3×5×6 within-participants RM-ANOVA with factors *Compensation* (off, on), *Block* (1-3), *Lag* (11 , 41, 74, 114, 164 ms), and *ID* (1.15, 1.76, 1.81, 2.15, 2.54, 3.02). Dependent measures were task completion time and number of errors.

The tracking study used a 2×3×5×2 within-participants RM-ANOVA with factors *Compensation* (off, on), *Block* (1-3), *Lag* (11, 41, 74, 114, 164 ms), and *Target Speed* (240, 320 u/s). The dependent measure was time on target.

Our hypotheses for the studies were:

H1. Tracking time on target would decrease with lag.
H2. Acquisition time would increase with lag.
H3. Acquisition errors would increase with lag.
H4. Lag compensation would reduce the effect of lag on tracking time on target.
H5. Lag compensation would reduce the effect of lag on acquisition time.
H6. Lag compensation would reduce the effect of lag on acquisition errors.

RESULTS

Outliers were removed from the data for both studies by removing the worst 1% of trials within each lag level for each participant (268 total trials removed, equivalent to one-tailed 3σ s.d.). We chose the one-tailed approach because it was impossible for a trial to be too fast in our system, but trials could be too slow if participants moved their hand from the mouse, rested, or asked questions during a trial.

Acquisition Study

Trial completion time included trials with misses. We chose to incorporate error rate into the time metric because this approach reflects the effect of misses in real FPS games.

Effect of Lag on trial completion time

ANOVA showed a significant main effect of *Lag* ($F_{4,60}$=246, $p < 0.0001$), as well as an interaction effect between *Lag* and *Compensation* ($F_{4,60}$=18.7, $p < 0.001$). In the non-compensated case, planned pairwise t-tests (Holm corrected) showed a significant difference between all levels of lag ($p < 0.0001$), with the exception of the 11 ms and 41 ms pair ($p = 0.25$). We therefore accept **H2**.

Figure 3 shows the effect of lag on trial completion time, both with and without compensation (black lines). The effect of 41 ms of lag was negligible compared to baseline, although it becomes substantial and approximately linear at higher levels. Trial completion time increased by 1.5%, 11.0%, 28.5%, and 52.1% compared to baseline at 41 ms, 74 ms, 114 ms, and 164 ms, respectively.

Effect of Compensation

The interaction between lag and compensation also suggests that compensation may have been effective at reducing the effect of lag. However, with compensation enabled, planned pairwise t-tests showed that the same pairs of lag had a significant effect on performance as in the non-compensated case (74 ms and higher). This means that our compensation technique was not able to eliminate the effect of lag. However, Figure 3 shows that compensation did decrease the effect, and follow-up paired t-tests (Holm corrected) confirm a significant improvement at 74, 114, and 164 ms ($p < 0.05$), both when comparing the compensated curve to baseline, and when comparing within the compensated curve at the 11 ms level. We therefore accept **H5**.

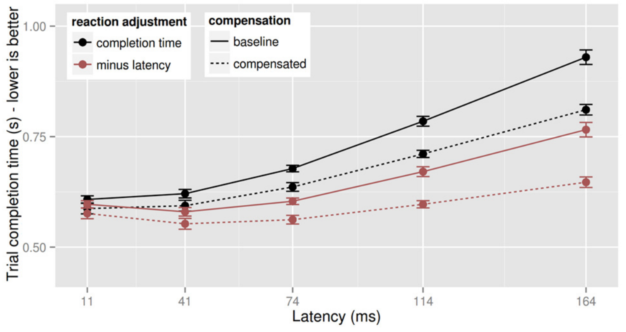

Figure 3: Effect of latency on targeting performance.

Figure 3 also contains dark red lines depicting trial completion time when the amount of latency is subtracted from completion time. With lag, the participant will see the target appear later than when it actually gets created within the game state, and therefore reaction time will be delayed by the amount of latency present. All subsequent figures will display non-adjusted performance (black lines).

Effect of ID

ANOVA showed a significant interaction between *Lag* and *ID* ($F_{20,300}$=8.87, $p < 0.0001$), as well as a three-way interaction between *Lag*, *ID*, and *Compensation* ($F_{20,300}$=2.15, $p < 0.001$). This suggests that lag has a different effect on acquisition performance depending on acquisition difficulty. Figure 4 shows that there actually was a performance decrease between 11 ms and 41 ms of lag at the lower three difficulties, confirmed by a followup paired t-test ($p < 0.001$), although not at higher difficulties ($p > 0.5$). It can also be seen that compensation worked for all *ID* levels.

Figure 5 shows the same *Lag*, *ID*, and *Compensation* factors, but with *ID* on the x-axis. Since the curves are approximately linear, it is appears that the acquisition task in the FPS environment follows Fitts' Law when angular distances are used. Note however the sharp increase in completion time between the very close second and third *ID* at higher latencies. These points are associated with a change in both radius and distance, reinforcing previous results [23].

Effect of Block

ANOVA showed a significant main effect of *Block* ($F_{2,30}$=6.76, $p < 0.05$), indicating that a practice effect was present. Trial completion time decreased by 1.5% in block 2 and 3.6% in block 3 compared to block 1.

Error rate

We also examined the effect of lag on acquisition error rate. ANOVA showed a significant interaction between *Lag* and *Compensation* ($F_{4,60}$=6.32, $p < 0.001$). Without compensation, paired t-tests show that error rate is lower at 41 ms than at 11 ms ($p < 0.01$), and that there is no significant difference between 11 ms and the other levels, so we must reject **H3**. With compensation enabled, paired t-tests show a significant decrease in error rate at 114 ms and 164 ms compared to 11 ms of lag, and no significant difference at 41 ms and 74 ms. We therefore accept **H6**. The decrease in error rate compared to the baseline suggests that our compensation technique could benefit from further tuning.

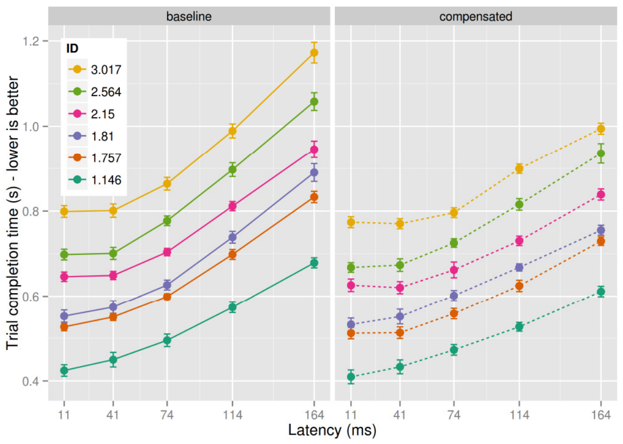

Figure 4: Effect of latency on targeting performance by *ID*.

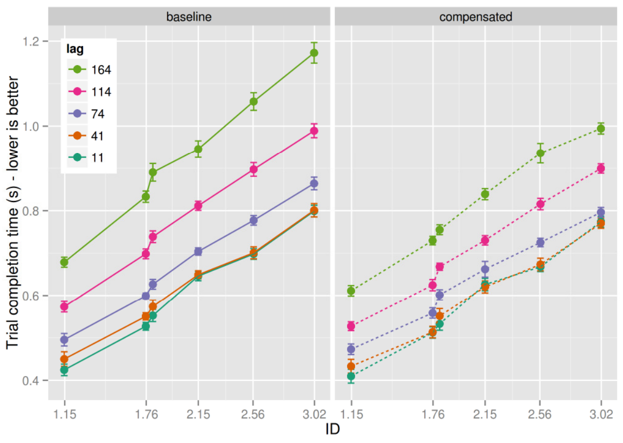

Figure 5: Effect of difficulty on targeting performance by latency and compensation.

Tracking Study

Performance in the tracking study was determined by the mean amount of time successfully spent tracking the target per trial, with maximum possible time being 5 seconds.

Effects of Lag on tracking time

ANOVA showed a significant effect of *Lag* ($F_{4,56}$=39.7, $p < 0.0001$), as well as a significant interaction between *Lag* and *Compensation* ($F_{4,56}$=46.2, $p < 0.0001$). With no compensation, planned pairwise t-tests (Holm corrected) showed a significant decrease in tracking time between each level of lag ($p < 0.01$). We therefore accept **H1**.

As shown in Figure 6, the performance decrease is approximately linear with increasing lag: 94.2%, 87.8%, 77.7%, and 67.3% of baseline latency performance at 41 ms, 74 ms, 114 ms, and 164 ms of latency.

Effect of Compensation

ANOVA showed a significant main effect of *Compensation* ($F_{1,14}$=22.6, p<0.001) and an interaction between *Lag* and *Compensation* ($F_{4,56}$=46.2, p<0.0001), meaning that compensation was effective at reducing the effect of lag. With compensation enabled, pairwise t-tests showed no significant difference in tracking time between any levels of lag (p > 0.2). T-tests also showed a significant difference between the baseline and compensated tracking times at all levels above 11 ms (p < 0.01), with no significant difference at 11 ms. We therefore accept **H4**.

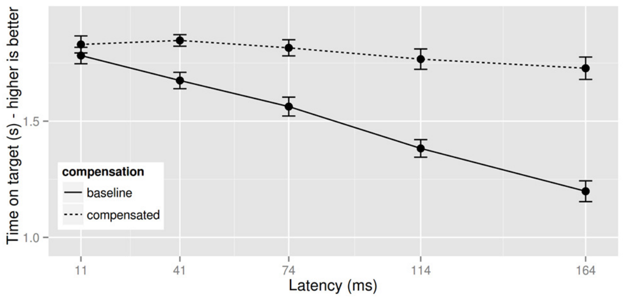

Figure 6: Effect of latency on tracking performance.

Effect of Speed

An interaction was found by ANOVA between *Lag* and *Target Speed* ($F_{1,14}$=4.40, p < 0.01), suggesting that lag affects performance differently depending on target speed. Figure 7 shows that performance impact of lag is greater at lower target speeds than higher target speeds.

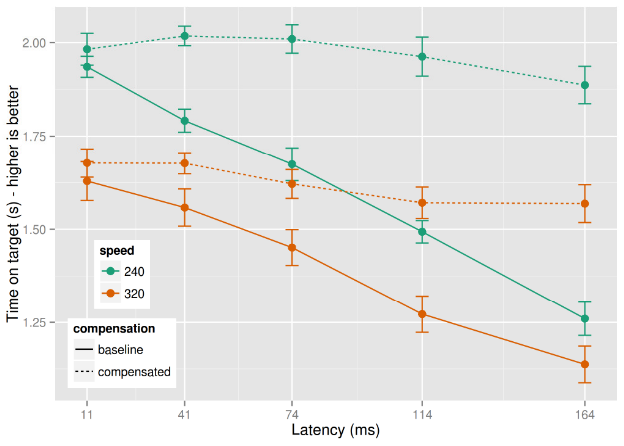

Figure 7: Effect of latency on tracking performance, by speed.

A significant three-way interaction between *Lag*, *Speed*, and *Compensation* was also shown by ANOVA ($F_{4,56}$=3.38, p < 0.05). The effectiveness of our compensation technique varies depending on the amount of lag and target speed, with it giving a slight boost compared to baseline performance at lower target speeds at 74 ms and 114 ms.

Effect of Block

ANOVA showed a significant main effect of *Block* ($F_{2,28}$=8.98, p < 0.001), suggesting that there was a learning effect present. Tracking time increased by 4.1% in block 2 and 6.5% in block 3, as compared to block 1.

Questionnaire Results

Surveys showed that participants could identify the presence of lag and its effect on performance at 114 ms of lag and higher (MWW test, p<0.05), and could not distinguish between lower levels. With compensation enabled, they felt that they performed equally well at all levels of lag.

DISCUSSION

Our studies provide clear answers to the three questions:

- *What is the prevalence and magnitude of local latency*? Our first study showed that a sample of real-world gaming setups have local latencies from 23 to 243 ms.
- *What are the effects of local latency on 3D aiming*? Both targeting and tracking performance were significantly degraded with local latency. Completion time for target acquisition increased significantly for lag above 41 ms (for higher difficulty targets) and above 11 ms (for lower difficulty targets). Time on target during tracking decreased significantly for lag beyond 11 ms.
- *Does compensation reduce the effects of local latency*? Compensation was very effective. The negative effects of latencies were reduced by up to 40% for target acquisition and eliminated altogether for target tracking.

Explanation and Interpretation of Results

The reasons why local latency causes performance to decrease are likely the same as those discussed for other types of latency – the surprising finding in our study, however, is that even latencies as low as 41 ms can cause significant performance problems for aiming. Few studies have suggested that human reaction time can be affected by latencies this small – although one recent study of ultra-low-latency touch screen hardware also found that latency above 10 ms had a negative effect on touch performance [11].

The difference in sensitivity between targeting and tracking may arise from the added requirements of the tracking task – during tracking, changes in the direction of target movement were observed to cause significant disruption, and direction changes occurred numerous times during each trial. Direction changes cause a double penalty in tracking – not only does it take longer to react to a direction change with lag, but the target has moved farther in the opposite direction.

As with the effects of latency, the reasons for the success of our compensation mechanisms also align with previous work. That is, the techniques worked well because they provided a kind of bridge across the delay-caused gap between the initiation of an action and the display of that action. For targeting, compensation enlarged the effective width of targets by the amount needed to make up for the overshooting that occurred due to latency; for tracking, aim dragging kept the crosshair on the target for the amount of time in which the target's direction change was invisible due to delay.

The difference in compensation effectiveness between targeting and tracking (compensation did not work as well in targeting) arises from the fact that we only compensated for overshooting errors, rather than lateral errors. We initially thought that latency would primarily cause overshooting, but we observed that lateral aiming errors were also common, and the presence of lag hampered the player's ability to adjust their aim through controlled feedback. Future work will examine techniques that take these transverse errors into account (e.g., bullet magnetism [25]).

Again, the surprising finding in our study of compensation techniques was the degree of effectiveness of the technique, even at the highest levels of local latency that are likely to be experienced in the real world. This means that the problem of local latency is one that could potentially be solved if compensation techniques can be successfully deployed.

Generalizing the results to real FPS games

Although our study was (necessarily) a controlled experiment rather than a real-world test in a real FPS game, there are several reasons why our results about the effects of latency will hold in realistic scenarios:

- We tested amounts of latency that are common in gaming systems, and covered a wide range of systems with different input and display devices in various games.
- We tested realistic task elements (targeting and tracking) that are ubiquitous in 3D shooter games. Although there are many differences between our study system and an actual FPS game (e.g., moving while shooting, other visual information), the core mechanics of moving the mouse to acquire or track a target do not change.
- The ways that targeting and tracking work in most FPS games is similar to out experimental tasks. FPS players typically aim for fixed targets in a single aim-and-shoot action, and moving targets typically move along a horizontal plane (e.g., the ground or a ledge) with most evasive action occurring as horizontal direction changes.
- We used an experimental setup that closely mimics a real gaming PC (fast mouse, graphics card, and display), and recruited participants with FPS gaming experience.

There are some differences that should be studied further in a real FPS environment. For example, in the study system the player could not move the character. While player movement is often used as part of tracking in FPS games, this restriction allowed us to focus specifically on mouse input for tracking tasks (although many FPS games encourage the player to stay stationary while firing in the form of reduced accuracy when firing while moving). However, we believe that differences between our testing environment and a real FPS environment will most likely increase the effects of lag. Player movement adds additional complexity when aiming and provides further opportunity for lag to interfere with the feedback processes. Likewise, the combination of movement and pressure to fire accurately encourages players to use the closed-loop feedback phase of aiming which is highly affected by lag due to the need for multiple adjustments with feedback delay between each one. Similarly, mechanics such as gun recoil and bullet spread used by some FPS games introduce an additional feedback loop into the aiming process where lag may have an even greater detrimental effect. Unlike real games, our experiment involved repeated simple aiming motions where participants could easily focus on the open-loop ballistic phase of motion.

Given enough time, players can get used to the latency of a particular system and notice it less. Players often claim that this adaptation reduces the effect of lag, and although that may be true to a small extent, we believe the adaptation effect is limited. Lag reduces the amount of time available to respond to game events. Planned acquisition motions such as open-loop ballistic motion with correctly predicted target locations may become latency-adapted to some extent, but errors are common and would negate adaptation. Likewise, tracking predictable targets may suffer less from lag, but any unexpected path deviations or errors in tracking would suffer from the closed-loop feedback-delay penalty imposed by latency. Our experiment gave players a chance to adapt to lag by making smaller, sequential changes in small steps rather than randomizing latency levels. Since there was no interaction effect between *Block* and *Compensation* in either task, it appears that players were sufficiently adapted.

Our study setup also supports generalization of the effects of compensation to real FPS games. Our compensation technique was designed to be general enough to apply to many common aiming scenarios. Two issues were not studied in our experiment: first, the effects of large required movements to acquire a target (e.g., real gameplay might require the player to perform a rapid 180 degree turn to aim at an enemy); second, the potential effects of distractor targets. Distractors that can appear near the intended target can make aiming assistance techniques less effective [25], even though the technique is designed to only affect a small angle around the targeting cursor. In real FPS scenarios, distractors may be less of an issue because players typically shoot at the nearest target. In addition, modifications to the compensation algorithm – such as determining the size of the effect based on cursor speed – may help to avoid errors.

There are some important challenges in deploying a compensation scheme in a real-world FPS. One main issue is the need to know the amount of local latency in order to set the amount of compensation. If the estimated latency is incorrect, compensation will either work ineffectively or provide an unfair advantage to the player. To accurately determine local latency, measurement tools should be used. These tools are already in use for latency-sensitive devices such as the Rock Band game controller and the Oculus Rift headset. However, to ensure fairness it would be better if latency estimates could be made directly from the hardware components that make up a game system. We envision that the required information could be archived in a database of component latency specifications, ideally created by manufacturers and from crowd-sourced gaming communities. The game could then require that players establish their latency before allowing compensation to be enabled.

Questions for further study

Are target size and target distance independent factors for ID in the presence of latency? So and Chung [23] suggest that target size and distance should not be considered a single effect of ID for tasks involving latency. We included two target conditions with similar IDs, but with different combinations of distance and size. With lag present, the closer and smaller targets had consistently worse performance than the targets with greater distance and size (but similar ID). The performance discrepancy with these similar IDs also increased with lag (Figure 5). This would suggest that smaller targets cause greater problems for local latency. However, this preliminary finding will need to be verified by a more targeted study as future work.

Comparison to latency in 2D tasks. Although existing studies of latency in 2D pointing tasks (e.g., [16]) use different methodologies, conditions, and latency levels than our experiment, it seems that latency may have effects of similar magnitude in 2D pointing as in 3D. However, studies on 2D target tracking under latency show a different scenario. In one study [20], tracking errors were only significantly greater than baseline at 170 ms of latency, as opposed to our results which show degradation from 41 ms.

CONCLUSIONS

This study provides the first empirical evidence quantifying the negative effects of local system latency on aiming in 3D game environments. Our results show that local latency as low as 41 ms caused significant and substantial degradation in aiming performance and that performance degradation increased as latency increased. Our results also show that significant amounts of latency prevail in real-world systems and that aim assistance can mitigate local latency. We show that problems caused by lag can be compensated for, reducing the problems for acquisition and removing them altogether for tracking. Our work shows that local latency is a real and substantial problem – but that games can mitigate the problem with appropriate compensation methods.

REFERENCES

1. Accot, J., and Zhai, S. Refining Fitts' law models for bivariate pointing. *CHI 2003*, 193-200.
2. Beigbeder, T., et al. The effects of loss and latency on user performance in Unreal Tournament. *NetGames 2004*, 144-51.
3. Braga Sangiorgi, U., et al. Assessing lag perception in electronic sketching. *NordiCHI 2012*, 153-161.
4. Chen, L., et al., An HCI method to improve human performance reduced by local-lag mechanism, *Interacting with Computers 19*, 2 (2007), 215-224.
5. Elliott, D., Helsen, W., and Chua, R. A century later: Woodworth's (1899) two-component model of goal-directed aiming. *Psych. Bulletin 127*, 3, 2001, 342.
6. Fritsch, T., Ritter, H., and Schiller, J. The effect of latency and network limitations on MMORPGs: a field study of Everquest2. *SIGCOMM 2005*, 1-9.
7. Ghez, C., & Krakauer, J. The organization of movement. *Principles of neural science 4,* 2000, 653-73.
8. Gutwin, C. The effects of network delay on group work in shared workspaces, *Proc. ECSCW 2001*, 299-318.
9. Isokoski, P., et al. Gaze controlled games. *Universal Access in the Information Society 8*, 4, 2009, 323-337.
10. Jörg, S., Normoyle, A., and Safonova, A. How responsiveness affects players' perception in digital games. In *Proc. Applied Perception 2012*, 33-38.
11. Jota, R., Ng, A., Dietz, P., and Wigdor, D. How fast is fast enough?: a study of the effects of latency in direct-touch pointing tasks. *CHI 2013*, 2291-2300.
12. Kopper, R., et al. A human motor behavior model for distal pointing tasks. *IJHCS 68*, 10, 2010, 603-615.
13. Leadbetter, R. *Console Gaming: the Lag Factor.* (2009) eurogamer.net/articles/digitalfoundry-lag-factor-article
14. Lee, B., Bang, H. A kinematic analysis of directional effects on mouse control. *Ergon. 56*, 11, 2013, 1754-65.
15. Looser, J., Cockburn, A. and Savage, J. On the validity of using First-Person Shooters for Fitts' law studies. *People and Computers 2005*, 33-36.
16. MacKenzie I., Ware C. Lag as a determinant of human performance in interactive systems. *CHI 1993*, 488-493.
17. Mandryk, R.L., and Gutwin, C. Perceptibility and utility of sticky targets. In *Proc. GI 2008*, 65-72.
18. Matejka, J., Grossman, T., and Fitzmaurice, G. Swift: reducing the effects of latency in online video scrubbing. *CHI 2012*, 637-646.
19. Pantel, L. and Wolf, L. On the impact of delay realtime multiplayer games. *Proc. NOSSDAV 2002*, 23-29.
20. Pavlovych, A., and Stuerzlinger, W. Target following performance in the presence of latency, jitter, and signal dropouts. *Proc. GI 2011*, 33-40.
21. Pavlovych, A., and Gutwin, C. Assessing target acquisition and tracking performance for complex moving targets in the presence of latency. *GI 2012*, 109-116.
22. Savery, C., Graham, T.C., and Gutwin, C. The human factors of consistency maintenance in multiplayer computer games. *CSCW 2010*, 187-196.
23. So, R., Chung, G. Sensory motor responses in virtual environments. *IEEE EMBS 2005*, 5006-8.
24. Stuckel, D., and Gutwin, C. The effects of local lag on tightly-coupled interaction in distributed groupware *CSCW 2008*, 447-456.
25. Vicencio-Moreira, R., et al. The effectiveness (or lack thereof) of aim-assist techniques in first person shooter games. *CHI 2014*, 937-46.
26. Ware, C., Balakrishnan, R. Reaching for objects in VR displays: lag and frame rate. *ToCHI 1*, 4, 1994, 331-356.
27. Wattimena, A.F., et al. Predicting the perceived quality of a first person shooter: the Quake IV G-model. *SIGCOMM*, 2006, 42.
28. Wilson, D. *Triple Buffering: why we love it.* Feb 2009. http://www.anandtech.com/show/2794/4.