Post-Render Warp with Late Input Sampling Improves Aiming Under High Latency Conditions

JOOHWAN KIM, PYARELAL KNOWLES, JOSEF SPJUT, BEN BOUDAOUD, and MORGAN MCGUIRE, NVIDIA

Fig. 1. Player view and key result from our experiment. *Left:* In traditional graphics pipelines, the generated image reflects the player input before rendering. *Center:* Late-warp updates the image based on late-latched player input. *Right:* Average task completion time shows a significant player performance penalty between 105 ms (No Warp) and 25 ms (Baseline) latency conditions. Error bars indicate the standard error of the mean of individual median scores (see Section 4). All warp methods tested mitigate this penalty.

End-to-end latency in remote-rendering systems can reduce user task performance. This notably includes aiming tasks on game streaming services, which are presently below the standards of competitive first-person desktop gaming. We evaluate the latency-induced penalty on task completion time in a controlled environment and show that it can be significantly mitigated by adopting and modifying image and simulation-warping techniques from virtual reality, eliminating up to 80% of the penalty from 80 ms of added latency. This has potential to enable remote rendering for esports and increase the effectiveness of remote-rendered content creation and robotic teleoperation. We provide full experimental methodology, analysis, implementation details, and source code.

CCS Concepts: • **Computing methodologies** → **Image-based rendering**; **Distributed algorithms**; • **Software and its engineering** → **Interactive games**; • **Networks** → **Network services**.

Additional Key Words and Phrases: latency, streaming, esports

ACM Reference Format:
Joohwan Kim, Pyarelal Knowles, Josef Spjut, Ben Boudaoud, and Morgan McGuire. 2020. Post-Render Warp with Late Input Sampling Improves Aiming Under High Latency Conditions. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 2, Article 12 (August 2020), 18 pages. https://doi.org/10.1145/3406187

[1] *Oracle* refers to rendering images with more recent information, e.g. view rotation and translation, while other game state is still delayed to simulate 80 ms of latency. This isolates the image warp implementation problem.

Authors' address: Joohwan Kim, sckim@nvidia.com; Pyarelal Knowles, pknowles@nvidia.com; Josef Spjut, jspjut@nvidia.com; Ben Boudaoud, bboudaoud@nvidia.com; Morgan McGuire, mcguire@nvidia.com, NVIDIA.

1 INTRODUCTION

Remotely-rendered, interactive, high performance graphics systems that stream video over a network are an important part of the graphics ecosystem. Some already-deployed use cases are cloud gaming services, cloud-hosted computer-aided-design/digital-content-creation software, robot teleoperation for remote surgery, autonomous vehicle fail-over, and drone piloting. In the latter cases, 3D rendering may not even be performed, and the scene will be instead communicated via "first-person" camera video and LIDAR depth streams.

These applications share the property that *task performance* is the primary goal, while visual fidelity is secondary. They also have the property that performance of tasks such as aiming and steering is degraded by network latency dominating the net motion-to-photon latency for the interaction loop.

We conducted a human-computer interaction experiment to evaluate latency-compensation techniques, originally developed for comfort in the virtual reality community, for improving aiming task performance in remote-rendering applications. We focus specifically on accelerating today's commercial game streaming services. We seek to make players using such services competitive with those rendering locally, for the first-person shooter (FPS) games that drive growth in the esports industry [29]. This paper bounds the effectiveness of latency compensation for task performance, evaluates how much of that effectiveness can be achieved using existing techniques, and presents guidance on how to research and design future systems.

Our focus is the addition of image warping for latency compensation. Image warping complexity ranges from simple camera view reprojection to temporally updating the animated contents. By updating a pre-rendered image based on more recent input, such as player movement and camera direction, perceived latency can be reduced. We describe system modifications to the streaming server and client to accommodate just-in-time late sampling of user input and post-render image warping. Our system makes latency adjustments to preexisting rollback code in the game server's hit detection model. We instrument a gaming system, including a software latency simulator and hardware latency measurement device.

Warping a previously-rendered image to a new viewpoint has been an active research topic for more than two decades (e.g., [3, 27]), and the resulting image quality varies widely with the algorithm. Instead of evaluating a single image warping algorithm, we experimentally bound the effect of the entire class of egomotion image warping for first-person aiming. We use three warp conditions: Rotation-Naive (**RN**), Rotation-Oracle (**RO**), and Translation-Rotation-Oracle (**TRO**).

TRO is the upper bound: a full view reprojection, compensating for more recent view Translation and Rotation from both mouse and keyboard inputs (Fig. 1 center bottom). E.g., if the player moves left, the image contents must move right. In practice this would involve a complex image transform with holes and disocclusions from parallax, discussed in Sec. 2. For this upper bound we instead use an Oracle[1], simulating the result.

Our lower bound is a Rotation warp using a naive approach of leaving pixels black if the corresponding view is beyond that of the pre-rendered image (**RN**, Fig.1 center top), without any guard band, depth map, motion vectors, etc., that can be applied to real world video and existing systems as well as rendered images. Existing image warping approaches discussed in Sec. 2 fall between these cases. We also separately evaluate the hybrid Rotation Oracle (**RO**, Fig.1 center middle) warp to distinguish the effect of ignoring translation from the use of oracle data. The image space operation of these warps are visualized in Fig. 2. From the perspective of the final on-screen image, Fig. 3 describes the state seen from different times depending on the warp condition.

Our main research contributions are the novel experiment and analysis. We analyzed 6,750 user trials with varying conditions and conclude:

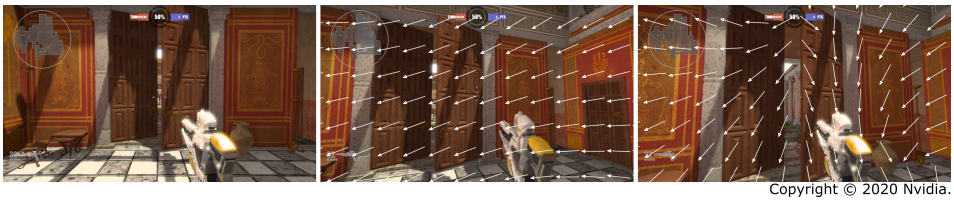

Fig. 2. Visualizing a reprojection in image space. *Left*: Pre-rendered image. *Center*: Rotation applied. *Right*: Translation and rotation applied. Note that the aim changes significantly and some pixels are not in the pre-rendered image.

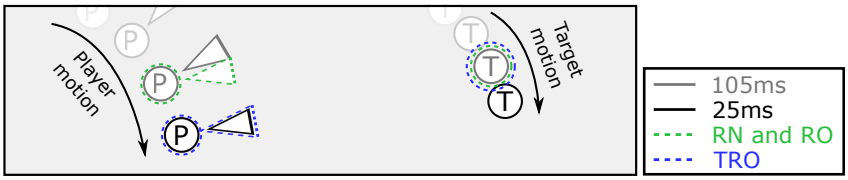

Fig. 3. Conditions **RN**, **RO**, and **TRO** are characterized by the latency on rotation and translation of the player. In this (top-down) illustration, the player is translating downward and rotating clockwise, and the target is translating downward. **RN** and **RO** present the latest player rotation but delayed player translation. **TRO** presents latest player translation and rotation. Target position is delayed in all three conditions (**RN**, **RO**, and **TRO**). See Sec. 3.4 for their implementations.

(1) Image warping with appropriate system modifications eliminates up to 80% of the aiming task performance penalty incurred by 80 ms of additional latency.
(2) The image quality of the warp has a weak effect on task performance.
(3) Including translation in the warp has a weak effect on task performance.
(4) The latency penalty is significant for easy tasks and increases with difficulty.

The graph in Fig. 1 (right) shows aiming task completion time versus warp conditions, where lower task completion time is better. The task completion time difference between the no-warp condition (red) and the baseline condition (black) is the latency-induced performance penalty for the task between 105 ms and 25 ms of egomotion-to-photon latency (in the absence of warping). All three warp methods reduced this penalty substantially.

A minor additional contribution of this work is the system design and open-source game experiment code. A natural conclusion of this work is the need for continued rendering research on efficient warping algorithms that preserve image quality for aesthetic purposes, as aesthetics are important to the user experience even though their effect on task performance in this case is small.

2 RELATED WORK

The latency from user input to visual output on monitors is a critical factor for player performance in multiplayer games, especially for aiming in fast FPS games [17, 38]. Network latency between the game client and server is also important for player experience, as evidenced by reduced play time with higher network latencies [6]. Latency affects player performance [11, 12] at a varying degree depending on genre. In general, games with shorter deadlines are more strongly affected by latency (e.g., FPS games such as Unreal Tournament [4]) in contrast to real time strategy games such as Warcraft 3 [10, 37]. Today, latency sensitive genres form a large part of the video gaming industry [29], and video gaming platforms strive to reduce latency [16]. This section describes

previous work to mitigate these two areas of latency. While our focus is hiding input to visual output latency, its interaction with other latency hiding approaches is important.

Hiding network latency is an established concept in traditional multiplayer gaming. A number of latency compensation techniques have been used to create more responsive gaming experiences and improve synchronization among the game states of internet-connected users. Globally synchronizing clocks, also called dead reckoning, improves extrapolation accuracy [1]. Changing game mechanics to be easier and have longer deadlines can compensate [6]. However this technique might not be suitable for skill-sensitive games such as FPS as they tend to have short deadlines [12]. Bernier [5] keeps a history of game state so that, from the game server's point of view, player actions can be verified and applied retroactively. *Rollback* means restoring previous game state and possibly re-simulating with new actions. Rollback hides latency and maintains the illusion of a consistent world for all players. Timelines [34] introduce a model for the physical implications and paradoxical edge cases. Measuring latency is important as even differences in animations, such as the first person camera and third person avatar, can affect perceived latency [15].

Many competitive games hide network latency using one or more of the techniques mentioned above. This leaves local system latency as the remaining challenge to player performance in local gaming [17, 25, 31]. Contributors to local system latency include choice of input peripherals, output device (e.g., pixel response time), computing components (e.g., CPU and GPU), and various optimizations in the game application. For examples of these optimizations we refer the interested reader to [16]. The throughput of game state simulation and presentation (e.g., higher frame rates and monitor refresh rates) benefit player performance as well, mainly because they reduce local system latency [38].

Cloud gaming is a focus and motivates this research as network latency is added to what is normally local system latency. Major commercial services include Playstation Now, Xbox Game Streaming, GeForce NOW, Stadia, and Steam Remote Play. In these systems, the round-trip latency between input ("motion") and final display of a frame based on that input ("photons") is sufficiently low to provide an enjoyable experience for many games; however, it is also sufficiently high to degrade player performance at game tasks such as aiming, and thus makes skilled competitive gaming and certain high-precision single-player experiences nonviable. Rollback alone cannot solve the latency problem of the network delay between input and game update and between game update and display. Unsurprisingly, measured input to output latency of these services are longer than local gaming scenarios [28].

Several researchers propose cloud gaming systems that can improve responsiveness. Moving cloud renderers closer to the client by deploying them on edge servers can improve network latency (at the time $\sim$ 30% of players had networking latencies above 80 ms to cloud services) [8, 9]. Predicting and sending multiple versions of future game outputs can reduce latency further. Outatime [22] renders multiple sequences of images based on possible future user inputs, and the client chooses a sequence to match what the player actually does. Outatime also uses image based rendering to adapt the chosen stream to the client's more recent and continuous view updates. However, this kind of speculative execution increases computational load and potentially streaming bandwidth depending on the implementation.

Using image based rendering to reproject a view is a known technique to create new images for different angles [23] and even positions [7]. Reprojection can reduce rendering costs, creating stereo views from a single output image or temporally upsampled animation [2, 13]. VR headsets use reprojection to hide latency [43]. Rendering is performed for a given view position and takes some amount of time, during which the viewer often moves. A cheap reprojection can update the image to correct for this change and reduce the perceived latency. The resulting image is still

delayed, but the motion to photon latency is reduced. Reprojection can be varied across the image if it is scanned out over time on certain VR displays [14].

Translational reprojection introduces more problems to solve. Viewer translation causes a parallax effect and disocclusion creates holes in the warped image that need to be filled. A common approach is using depth or velocity buffers to transform image pixels in 3D. Oculus recently introduced asynchronous spacewarp 2.0 [30], which requires depth data from games. Building spatial data structures [41] and using proxy geometry [32] can improve performance. Holes and missing information behind foreground objects need filling or additional data [19, 35, 36]. Non-solid geometry, transparency, and particle effects are composites from multiple depths. Specular highlights and reflections have yet more complicated image space transforms [24].

Translational and rotational warp accounts for just view movement, but there are also moving and animated objects in the image which are needed for a full warp. With sufficient rollback this may not matter for the outcome of some interactions, but immediate feedback can be important. Small deltas can be handled with per-pixel velocity vectors [2]. Another option is rendering and compositing some content locally [18, 20]. A simple example of this is locally rendering a muzzle flash and weapon projectiles before they are created and rendered remotely.

Games today demand tighter integration with latency hiding techniques for the new era of cloud gaming. Late-warp is highly desirable because it can realize the low latency sought by gamers. However, there are many ways to implement late-warps, and each method comes with its own pros and cons in terms of user performance, image quality, and implementation complexity. We leave improving late-warp techniques for future work and instead first explore the results from a user performance perspective — the primary concern of competitive gamers.

Although Outatime [22] showed the benefit of late-warp on user performance, our approach differs from theirs in a sense that we seek generalizable answers. First, we design the conditions in our user study by identifying game components whose latency can be independently varied by late-warp. Second, we focus on low-level aiming performance which is common to most FPS games as opposed to tasks involving higher-level game sense which vary depending on game (e.g., stage clearing used in Outatime [22]). Third, we varied latency while fixing frame rate at 60 Hz, in contrast to Outatime where latency was controlled but frame rate was left uncontrolled.

Our study provides specific guidance for designing the next generation cloud gaming, or any other interactive graphics system that can benefit from reducing latency using late-warp.

3 METHODS

We measure the effect of of late-warp on user performance in FPS game play. Specifically, we ask which user input and visual information is latency-critical and affects aiming performance. We measure user performance under various in-game conditions, where we vary latency related to player input and game states while fixing frame rate at 60 Hz.

3.1 Subjects

Nine experienced gamers participated as test subjects (age 21 – 34, 1 female and 8 males). Our subjects were habitual gamers who played at least 10-20 hours per week (eight subjects with at least 20 hrs/week and one subject around 10 hrs/week). All subjects gave informed consent, and the experiment was conducted in accordance with the Declaration of Helsinki.

3.2 Task and Stimulus

Subjects performed an aiming task in a simple FPS game (FPSci [39]). In the game, the player and a target were placed in a square room. Two semitransparent and indestructible walls with a small gap between them were placed between the player and target (Fig. 5). While the walls

Fig. 4. The classic narrow gap choke point, as seen in *Counter-Strike: Global Offensive* [42] and *Valorant* [33].

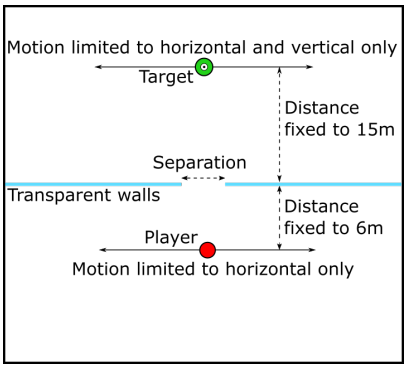

Fig. 5. Geometric layout of our experiment. Two semitransparent walls were placed along the plane bisecting the virtual room. The player and target were on the opposite sides of the walls, and their movement was limited to parallel to the walls.

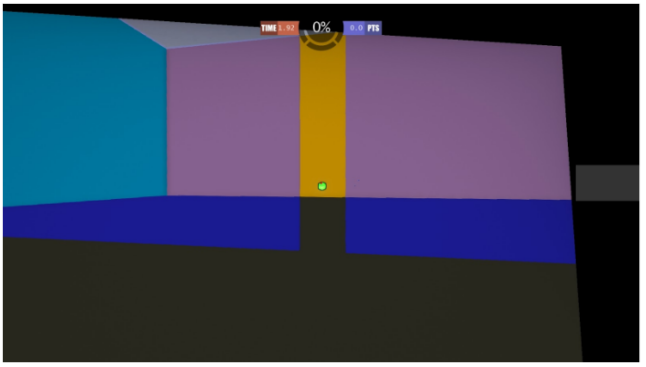

Fig. 6. Missing guard band artifact in **RN** shown as the black area around the top and right edge. 2D user interface elements (progress indicator at the top and latency measurement box at the right) were added after late-warp and are thus unaffected by the artifact.

were semitransparent, the gap between them was intended to approximate situations in classic FPS games where the player shoots through narrow gaps (Fig. 4). The player and target were 6 m and 15 m away from the walls respectively. The movement of player and target were restricted to directions parallel to the walls (the arrows in Fig. 5). The scene was rendered with a horizontal field of view of 103° at 1920x1080 resolution.

Each trial started by resetting the player position and orientation to their initial values, the midpoint of player travel space (the red dot in Fig. 5), looking through the midpoint of the wall separation, with a line of sight parallel to the ground. A dummy target (icosahedron with diameter of 0.66 m in game or 24 pixels on screen) was placed along the player's initial aiming point. A trial began by clearing the dummy target, spawning a moving target (icosahedron with diameter of 0.66 m in game or 24 pixels on screen) near the dummy target with a slight vertical offset (see Sec. 3.3.4). The task was to clear the moving target as quickly as possible. Players had to align the central aim point with the moving target and click the left mouse button to reduce the target's health. The target was cleared after a certain number of hits (see Sec.3.3.3), ending the trial.

Warp condition	Description	Egomotion latency (ms)		Game state latency (ms)
		Rotation	Translation	
105ms	No warp, cloud gaming latency.	105	105	105
RN	Rotation-Naive (has artifacts).	25	105	105
RO	Rotation-Oracle.	25	105	105
TRO	Translation-Rotation-Oracle.	25	25	105
25ms	No warp, local gaming latency.	25	25	25

Table 1. We measured aiming performance in five warp conditions. The two edge conditions do not use late-warp and have latency similar to those of typical cloud and local gaming scenarios (105 and 25ms). Three late-warp conditions include one representative late-warp implementation with implementation-specific artifacts (**RN**) and two idealized conditions to provide upper bound estimates of performance benefit.

3.3 Conditions

We focus on a range of warp conditions, measuring user performance improvement for each. In addition, we also varied scene configuration (wall separation), weapon behavior, and target motion to examine the effect of each on aiming performance under various conditions.

3.3.1 Warp Condition. We designed five warp conditions (Table. 1). The conditions differed by latency, with the frame rate fixed to 60 Hz.

105ms is a baseline condition in which the average latency is about 105 ms, similar to that of today's typical cloud gaming [28].

Rotation-Naive (**RN**) applies rotational late-warp on a pre-rendered game scene. No guardband is rendered and missing pixels are displayed as black (Fig. 6). Players experienced responsive rotational motion but delayed translational motion and game events (e.g., firing animation, placement of miss decals on the wall, explosion of targets, etc), and guard band artifacts were noticeable. We expect this condition to serve as the lower bound estimation of player performance benefit due to late-warp.

Rotation-Oracle (**RO**) is an emulated rotational late-warp; we render using a delayed version of game state and camera translation, but with the latest rotation input. The player experience is the same as in **RN** in regard to latency, but without the guardband problem. We expect this condition to serve as the upper bound estimation of player performance benefit due to rotational late-warp.

Translation-Rotation-Oracle (**TRO**) is another emulated late-warp; in a delayed version of the game world, we place the camera using the latest translation and rotation input, and render a new image from this position. Players experienced responsive rotational and translational movement but delayed game events. We expect this condition to serve as the upper bound estimation of player performance benefit due to late-warp.

25ms is the second baseline condition mimicking today's typical (local) desktop gaming latency. This condition also serves as the ground truth measurement of low-latency (25ms) rendering in our experiment, though lower latency may be possible.

Any improvement to (or degradation of) aiming performance when compared to **105ms** represents a benefit (or detriment) of local versus cloud gaming. Comparing **RN** and **RO** gives an upper bound on the effect of **RN**'s implementation-specific visual artifacts on aiming performance. Comparing **RO** and **TRO** demonstrates the effect of an 80 ms latency difference between rotation and translation on aiming performance. Comparing **TRO** and **25ms** shows the effect of delayed game state on aiming performance.

3.3.2 Wall Separation. The inconsistent latency between translation and rotation in **RN** and **RO** can hurt aiming performance when players change their translation. To test this, we indirectly controlled the amount of translation by changing the size of wall separation to 10, 1, and 0.5 m, where the target moved to behind the wall more often when the separation is narrower. When a target moved behind a wall, subjects had two choices: either wait until the target came out from behind the wall, or move to a position with a clear path to hit the target through the wall separation. Because the task was to destroy the target as quickly as possible, we anticipated the players would move more when the separation was narrower.

3.3.3 Weapon Type. The effect of refresh rate and latency on aiming performance can depend on how the weapon fires [38]. We tested whether the effect of warp conditions depends on weapon types by using two weapon types. The discrete hit weapon required a new mouse click for each shot and had a cool-down period of 0.5 s, meaning the user had to wait half a second after each shot before another could be attempted, and required two hits to clear the target. Subjects tended to spend the time between shots observing target motion, then made a quick mouse motion to attempt a hit at the predicted target location (this action is called a 'flick' among gamers). The continuous weapon would fire repeatedly while the mouse button was depressed at an interval of 0.1 s, and required ten hits to clear the target. In contrast to the behavior with the discrete weapon, subjects tended to track the target at all times with the continuous weapon.

3.3.4 Target Motion. When target motion is simple and predictable, players have more time to plan actions in advance. Inconsistent latency for translation and rotation in the player view may degrade aiming performance more when the target motion is more unpredictable and complex. We tested this by providing two types of target motion: predictable and unpredictable.

Predictable targets spawned 0.4 m (15 pixels on screen) below the dummy target and made a horizontally oscillatory movement along the same trajectory (± 3.5 m in game or ± 127 pixels on screen) at the same frequency (1 / 4.5 s). Unpredictable targets spawned 0.5 m (18 pixels on screen) above the dummy target and floated within a 2D bounding space ([-5, +5] m horizontal and [-1.7, +2.0] m vertical, with respect to the target spawn position) while randomly changing motion speed ([3, 5] m/s) and direction at a random period of time ([0.8, 1.5] s). When reaching the bounds of the defined space, the unpredictable target bounced by reflecting its motion about the bound normal and continued moving.

3.4 Implementation

In our experiment we simulate latency and late-warp in a self contained application. In a real world system, implementing late-warp would require the following components:

- Camera metadata: projection and per-frame view matrices.
- Larger than normal camera field of view for a guardband.
- Separate rendered output for the game world and camera-locked objects (e.g., the HUD, menus and weapon view model), to avoid some complexities of a general solution for transparency and disocclusion.
- Depth images and/or per-pixel velocity vectors if translational reprojection is needed.
- The late-warp operation on the game world followed by compositing camera-locked objects.
- Integration with game state roll back. Warping changes what the client displays and the game mechanics must be aware of this. For example, hit detection should be performed for an updated aiming direction.

Adding late-warp to a game can be invasive, requiring source-code access to the game engine, and the investment of time and resources needs to be justified by the possible benefits.

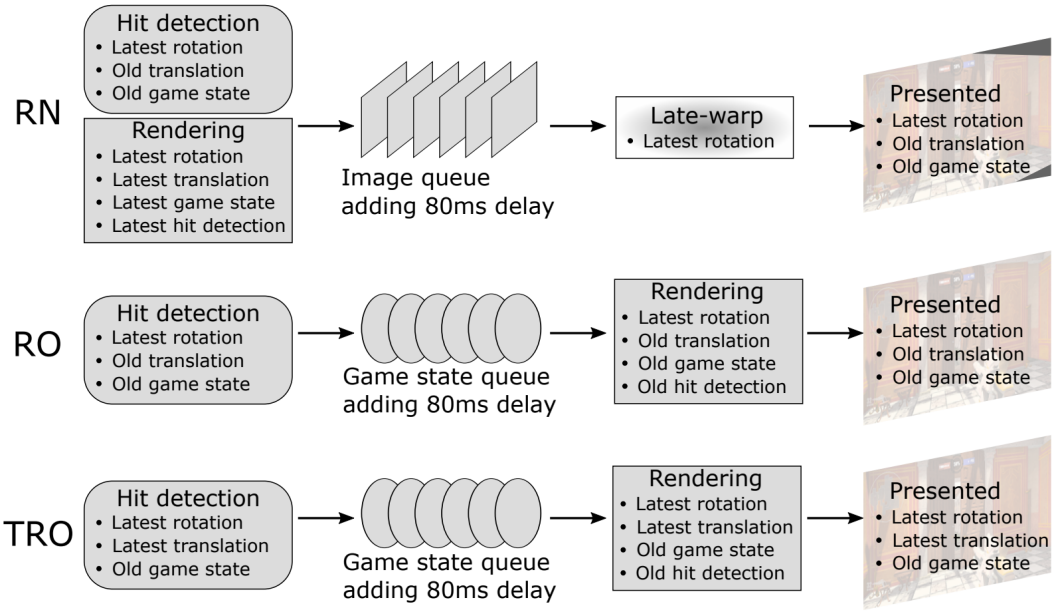

Fig. 7. RN implemented rotational warp by delaying image presentation, and RO and TRO simulated the respective warp methods by delaying game states. Rectangles with sharp edges represent operations on images, and rectangles with rounded edges represent operations on game states. See Sec. 3.4 for details.

In this study we measure and bound the player performance benefit by simulating latency and late-warp in a local system. We modified an open-source FPS game [40] and added conditions without added latency (**25ms**), with latency (**105ms**) and with latency and simulated late-warp (**RN**, **RO**, **TRO**). The modifications for this work can be found in the `hpg2020` tag on the FPSci Github [40].

25ms presented rendered images immediately. **105ms** delayed presenting rendered images by passing them through a queue, taking 80 ms and giving a total 105 ms average latency. Fig. 7 illustrates how we implemented the three late-warp conditions using image or game state queues. **RN** warped 80 ms old images with just the camera rotation delta from the time rendered to the current time. To do this, a perspective transform was applied to a full screen quad with the old image mapped as a texture. Rollback was implemented with state queues, in which game events, such as firing, were evaluated based on the state in the presented image. In this case, the latest camera rotation and the 80ms-old game state and camera translation. Since the rendered result is still delayed, the visual outcome of the evaluation was also delayed. **RO** achieved the same effect but rendered without delaying images. It used the latest mouse rotation input combined with the game state and camera translation from 80 ms beforehand. The result of game events were delayed similarly to **RN**, but due to the game state queue. **TRO** also rendered immediately and with an 80 ms old game state, but used both the latest camera rotation and translation (i.e. from recently sampled mouse position and keyboard inputs).

We ran our warp conditions on three PCs (Intel Core i7-9700k @ 3.6 GHz, 32 GB of RAM, RTX 2080 Ti). Subjects were seated at a PC, about 16 inches away from a 25 inch LCD monitor with a resolution of 1920x1080. We used a click-to-photon latency logger similar to the technique used by Spjut et al. [38] and confirmed that latency levels were correctly reproduced as designed (Table

1). We verified that for our selected mouse (Logitech G203), click-to-photon vs motion-to-photon latency were within 1 ms of each other. This represents a small fraction (1-4%) of the 25-100 ms end-to-end latency. As such, we assumed that click-to-photon latency can be used as a valid proxy for motion-to-photon latency in our system.

3.5 Procedure

The subjects completed data collection for one visualization condition per visit. Most subjects made only one visit per day. If multiple visits had to occur in a single day, then the consecutive visits were separated by at least an hour.

On the first day, we gave a 30 minute orientation where the amount of latency applied to game animation and player movement in each experimental condition were disclosed. Informing subjects about latency manipulations was a consciously made decision. Our goal was to predict the effect on actual gamers, who will be aware of the latency compensation mechanism if present. Thus, we disclosed it to our subjects in order to maintain consistency and accurately measure the effect on gamer performance. However, we did not mention the experimental hypothesis nor provided further information after the orientation. Subjects were instructed to perform at their best in all conditions. The ordering of conditions was pseudo-randomized per subject using a Latin Square for counter-balancing. While most subjects were able to identify **RN** and **105ms** conditions (**RN** was very obvious due to the guardband artifact, and **105ms** was noticeable due to high latency), **RO**, **TRO**, and **25ms** conditions were subtle and subjects did not clearly identify them.

There were 12 sessions in one condition. The first six sessions used the discrete weapon, and the second six sessions used the continuous weapon. With each weapon type, subjects first familiarized themselves with the warp condition and weapon type by performing three training sessions, which had wall separations of 10 m, 1 m, and 0.5 m, respectively. Subjects then performed three real data collection sessions, repeating the three wall separations in the same order. Each session contained 25 trials consisting of 15 unpredictable-motion targets and 10 predictable-motion targets (trials with predictable-motion targets were fewer because subject performance was expected to be more consistent due to the predictability). Completing one session took less than 3 minutes. Completing one warp condition took between 15 and 30 minutes. A publicly visible scoreboard was used to motivate players through friendly competition.

4 RESULTS

The design of our study was within-subject measurements using factors of warp condition, wall separation, weapon type, and target motion. For each factor, we performed repeated measures ANOVA on individual subject scores to examine its effects and interactions with the main factor of interest (the warp condition). We then applied pairwise t-tests between warp conditions per each experimental condition. For all subsequent plots, each data point represents the mean of individual subject scores, which were calculated as the median of all the scores in the corresponding condition, and each error bar represents the standard error of the mean. Table 2 provides the mean of individual scores and the standard error of the mean for all the conditions considered in the current section.

4.1 Effects of Warp Conditions

Figure 8 shows task completion time when aggregated across the five warp conditions. There was a significant main effect of the warp condition ($F_{4,32} = 22.9, p < 0.001$) on task completion time, where the F-statistic represents the ratio of between-condition variance and within-condition variance, and the p-value represents the level of significance of the effect we tested[26]. Pairwise t-tests indicate that **105ms** is the only warp condition that is significantly different from other

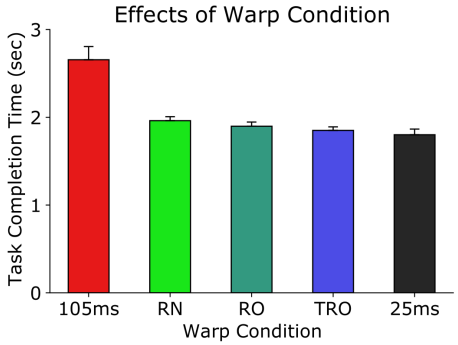

Fig. 8. Effects of warp condition on aiming performance. Bar height is the mean of individual median task completion time in the corresponding warp condition, aggregated across all other factors, and error bars are the standard error of this mean.

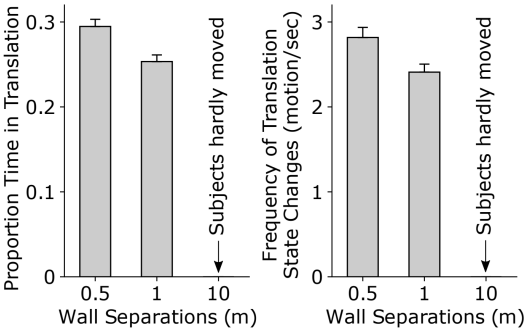

Fig. 9. Effects of wall separation on translational movements made by subjects. With narrower wall separations, subjects spent longer time in translation and more frequently made changes in translational state. Error bars represent the standard error of the mean across subjects.

warp conditions, namely **TRO** ($p < 0.005$), **RO** ($p = 0.001$), **RN** ($p < 0.01$), and **25ms** ($p < 0.001$). The difference among the other four conditions was not statistically significant.

The trend was monotonic in favor of more involved warp conditions: aiming performance improved as late-warp provided more information with short latency (**RN** > **RO** > **TRO** > **25ms**). Despite the relatively small differences in these four conditions that were not always statistically significant, this trend was observed quite consistently in the subsequent analyses.

The difference between **105ms** and **25ms** gives the performance detriment due to high latency. To quantify the aiming performance benefit in each condition, we compared this value against the difference between each warp condition and **25ms**. The three warp conditions were eliminating most of the performance difference between **105ms** and **25ms**; 81.3% (**RN**), 88.7% (**RO**), and 94.2% (**TRO**).

Guard band artifacts in **RN** did not significantly hurt aiming performance when compared to **RO**. The artifacts were clearly visible and reported by every subject. Nonetheless, subjects overcame the perceptual disturbance and performed almost equally for both conditions.

4.2 Effects of Wall Separation

As intended, narrower wall separation corresponded with subjects spending more time in translation ($F_{2,16} = 92, p < 0.001$) and changed the state of translation (i.e. stationary, moving to the left, and moving to the right) more frequently ($F_{2,16} = 193, p < 0.001$) (Fig. 9). Subjects hardly translated at the wall separation of 10 m but made translational movements at the two narrower separation levels. The difference between player translation for wall separations of 1 m and 0.5 m was relatively small, but the differences were significant ($p < 0.005$).

Figure 10 (left) shows task completion time for each visualization condition versus wall separation distance. We observed a statistically significant main effect of wall separation ($F_{2,16} = 350, p < 0.001$) and interaction between wall separation and warp condition ($F_{8,64} = 4.59, p < 0.001$). The **105ms** condition resulted in the longest task completion time at all wall separation levels ($p < 0.05$). The other four conditions did not significantly differ from each other with only one exception between **RN** and **25ms** at the wall separation of 1m ($p < 0.05$). At all the wall separation conditions, the

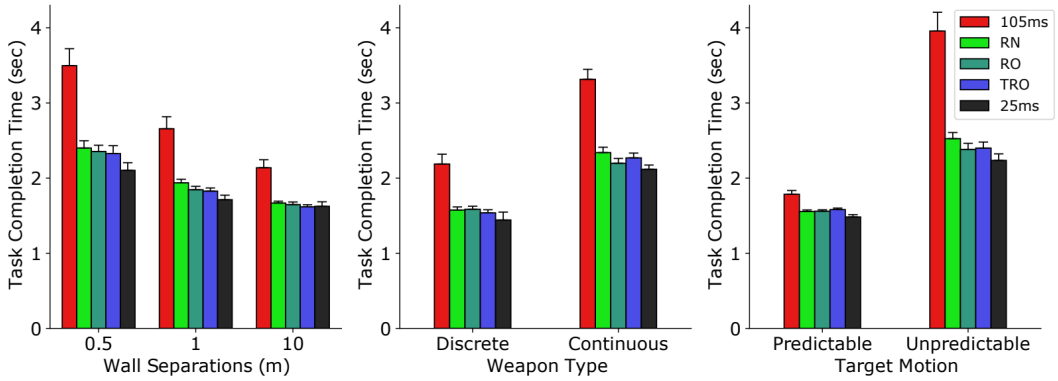

Fig. 10. Effects of wall separation (*Left*), weapon type (*Center*), and target motion predictability (*Right*) on aiming performance. **RN**, **RO**, and **TRO** yielded aiming performance similar to that in **25ms**. Error bars represent the standard error of the mean across individual median scores.

		Task completion time per warp condition (seconds)				
		105ms	**RN**	**RO**	**TRO**	**25ms**
Overall		2.66 ± 0.16	1.96 ± 0.05	1.90 ± 0.05	1.85 ± 0.04	1.80 ± 0.07
Wall separation	0.5m	3.50 ± 0.24	2.40 ± 0.10	2.35 ± 0.09	2.33 ± 0.11	2.10 ± 0.11
	1m	2.66 ± 0.17	1.94 ± 0.05	1.85 ± 0.05	1.83 ± 0.04	1.71 ± 0.06
	10m	2.14 ± 0.11	1.67 ± 0.03	1.65 ± 0.04	1.62 ± 0.03	1.63 ± 0.06
Weapon	Discrete	2.19 ± 0.14	1.58 ± 0.04	1.59 ± 0.04	1.54 ± 0.04	1.44 ± 0.11
	Continuous	3.32 ± 0.14	2.34 ± 0.08	2.20 ± 0.07	2.27 ± 0.07	2.12 ± 0.06
Target motion	Predictable	1.79 ± 0.05	1.56 ± 0.02	1.56 ± 0.02	1.58 ± 0.02	1.48 ± 0.03
	Unpredictable	3.96 ± 0.26	2.53 ± 0.09	2.38 ± 0.09	2.40 ± 0.08	2.24 ± 0.09

Table 2. Player performance measured per warp condition while varying wall separation, weapon type, and target motion predictability. Player performance is estimated as the mean across individual scores, the median of all the scores in the corresponding condition. The uncertainty bounds are the standard error of the mean. Late-warp conditions eliminated most of the performance penalty caused by 80 ms of additional latency.

three late-warp conditions eliminated 76.2% or more of the performance degradation between **105ms** and **25ms**.

4.3 Effects of Weapon Types

Figure 10 (center) compares aiming performance for the two weapon types tested. We observed a significant main effect of weapon type ($F_{1,8} = 184, p < 0.001$), and there was a significant interaction between weapon type and warp condition ($F_{4,32} = 4.64, p < 0.005$). With the discrete weapon, pairwise t-tests show that task completion time in **105ms** was significantly longer than all other conditions ($p < 0.005$). There were no other warp condition pairs that were significantly different. With the continuous weapon, task completion time in the **105ms** condition was significantly different from all other warp conditions ($p < 0.005$). In addition, the difference between **25ms** and two late-warp conditions were also significant, namely **RO** ($p < 0.05$) and **TRO** ($p < 0.05$). The three warp conditions were, again, not significantly different from each other. Over both weapon

types, the three late-warp conditions eliminated 80.8% or more of the performance degradation between **105ms** and **25ms**.

4.4 Effects of Target Motion

Figure 10 (right) compares aiming performance with the two levels of target motion predictability. We observed a significant main effect of target motion ($F_{1,8} = 354, p < 0.001$), and a significant interaction between target motion and warp condition ($F_{4,32} = 29.8, p < 0.001$).

We performed pairwise t-tests between warp conditions per target motion type. For predictable target motion, task completion time in **105ms** was significantly different from that in **RN** ($p < 0.05$), **RO** ($p < 0.01$), **TRO** ($p < 0.01$), and **25ms** ($p < 0.005$). Task completion time in **25ms** was also significantly different from that in **RO** ($p < 0.05$) and **TRO** ($p < 0.05$). When target motion was unpredictable, **105ms** differed from all four other conditions ($p < 0.005$). All other comparisons were not significantly different.

When target motion was predictable, the three warp conditions reduced task completion time by 76.1% (**RN**), 74.5% (**RO**), and 67.4% (**TRO**) of the difference between **105ms** and **25ms**. These values seem relatively small compared to what we have observed so far. The average task completion time between **105ms** (1.79 s) and **25ms** (1.48 s) conditions was small (0.31 s) when target motion was predictable. For reference, the presentation of game state was delayed by about 80 ms in the three warp conditions (see 3.3), which is about a third of that. The three warp conditions resulted in task completion time that was slower by 72.1 ms (**RN**), 77.0 ms (**RO**), and 98.3 ms (**TRO**) with respect to **25ms**. The delayed presentation of game state accounts for most of the performance detriment in the three warp conditions compared to **25ms**.

In contrast, when target motion was unpredictable, the three late-warp conditions eliminated 83.2% or more of the performance degradation between **105ms** and **25ms**.

5 DISCUSSION

We found that applying late-warp can eliminate most of the performance degradation caused by high latency. Even a naive implementation of rotation-only late-warp eliminated about 80% of this performance degradation. In addition, we tested three more warp conditions that give upper bound estimates of expected performance benefits. Solving guard band artifacts (**RO**), adding translational late-warp (**TRO**), and using game state prediction (**25ms**) are likely to provide additional benefit, but these improvements are smaller than that provided by rotational late-warp. This remained true even when players were encouraged to frequently make translational movements. This is great news in terms of computational efficiency because rotational late-warp is inexpensive and can easily give good visual results.

Rotation and translation may have different latencies due to different input device delays (i.e. mouse for rotation, keyboard for translation). Our experiment showed that rotation latency reduced aiming accuracy more than translation latency did, but temporal consistency between mouse and keyboard did not matter significantly. This may suggest that humans are able to learn how to deal with different latencies between mouse and keyboard and plan appropriately.

The differences in player performance between **105ms** and **25ms** suggest that low latency visual feedback for mouse input controlling rotation is an important aspect of aiming ability. Rotation from mouse input has extensive degrees of freedom as it can vary position along with all of its derivatives (i.e. velocity, acceleration, jerk).

Modern game graphics can be highly sophisticated. As examples of realistic challenges, almost all modern FPS games have HUD (heads up display) elements, most of which must stay out of the late-warp pipeline while the world view is being reprojected. To make the problem more complicated, many HUD elements are semitransparent and need to be blended with the content

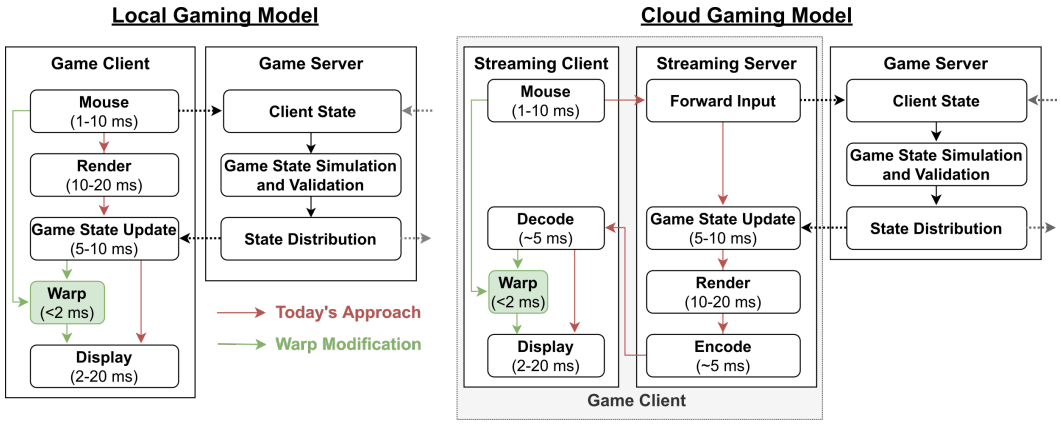

Fig. 11. Late-warp fits in as a modification of the standard gaming models and can substantially reduce motion-to-photon latency for some aspects. *Left:* Modification to a local gaming model. *Right:* Modification to a cloud gaming model. The red and green arrows indicate the steps comprising motion-to-photon latency in today's approach and warp modifications respectively. Note that while input prediction could be used, it adds error, and thus we only evaluate late input latching in this work.

behind them. Even worse, some HUD elements need to stay with the player view but some do not (and hence may need to be reprojected together with the view). The player view model adds more challenge; it must stay relatively fixed in the view but its appearance often changes according to external lighting. Applying rotational late-warp without breaking these elements is not trivial. For example, the HUD elements, mini-map, and weapon model stay fixed on the screen regardless of the warped 3D world in Figs. 1 and 2.

We studied late warp in the context of a simulation of added latency to a locally running game. The reason for the amount of latency we added comes from the additional latency added by the network links as well as video encoding and decoding added to the game client as shown in Fig. 11. While our experiment focused on the use of late-warp to reduce the performance penalty from large additional latency, late-warp may improve aiming performance in both local and cloud gaming situations.

Quantifying aiming performance benefits in an actual game is a topic for future study. Our experiment dealt with simple graphics and weapon dynamics, but how should we deal with complex weapon dynamics and projectile animation, and how will that affect aiming performance? Our experiment showed both player and target motion characteristics affect the benefit of late-warp, but how big will the benefit be for actual player and target motion in competitive games? Finding answers to these questions is only possible by implementing a warp method in a full game. It is a challenging task, but one that we believe to be worth pursuing as the player aiming performance benefits of late-warp can be substantial as shown in our results.

This study only considered first-person aiming and egomotion. Late-warp may benefit other genres such as third-person shooter, real-time strategy games, and other genres that involve continuous manipulation of motion. Whether our finding extends to these other genres is to be investigated.

The game state evaluation policy employed in our experiment deserves attention. We not only provided warped images to players, but also evaluated player actions in the game based on the game state of what the players saw. It is likely that our game state evaluation played an important role in compensating for the player performance penalty, although experimental verification is

needed to confirm it. To implement such feature in internet-connected games, any client performed late-warps needs to be included as one of the valid game states in the distributed world simulation. This can increase the possibility of cheating in cloud gaming, and hence additional attention is required for security in such situations.

Late-warp may be an attractive option for local gaming as well. Action games are popular, even among casual gamers. Many single-player adventure games such as Assassin's Creed or Tomb Raider are action-oriented, but often have high latency because developers focus more on high-fidelity rendering. Warp techniques may allow these games to provide responsive action while not compromising image quality. On the other side of the performance-fidelity trade-off, most competitive action games today implement relatively low rendering quality for many reasons, one primary reason being to keep latency low. Warp techniques may enable high-fidelity rendering for competitive games without increasing latency. In summation, we foresee late input sampling and post-render warping as a durable improvement for graphics quality and player experience in competitive and casual games alike.

Assuming warps are implemented in FPS games, will the importance of esports equipment (e.g., GPU and monitor) disappear? Probably not. Recall that throughput can be one fundamental limiting factor for latency. For example, a 60 Hz monitor imposes the average latency of 8 ms that cannot be eliminated by any means. The same is true when the rendering rate of the GPU is limited to 60 Hz. With the warp is done at the last moment, just before the presentation of the rendered scene, the throughput of the monitor and GPU again becomes the limiting factor in lowering the effective latency.

Day by day, more and more data and computing activities are moving to the cloud. Our experimental results imply a useful message for any interactive graphics applications that require internet connections: responsive local rendering or correction can significantly increase user performance. This is especially true if the affected task involves multiple action-perception cycles as with the aiming task in our experiment. Non-gaming examples include drawing on teleconference applications or typing, interactive documenting applications, and remote desktops [21]. We emphasize our finding here as a general principle because our experiment demonstrated its striking importance in the gaming context: 80 ms of latency could slow down task completion time by more than a second, and proper yet cheap local correction eliminates most of this detriment.

6 CONCLUSION

We have explored the effects of late-warp latency mitigation techniques on abstract first-person shooter style aiming tasks. Cloud gaming was a focus, where the latency that late-warp is capable of hiding is typically high. A user study was performed using a custom application to simulate latency in a controlled environment. Results show that even the most basic implementation eliminates up to 80% of the penalty from 80 ms of added latency. The oracle warp cases, **RO** and **TRO**, give modest improvements thereafter and in general show similar user performance to the low latency case.

Real-world implementation is nontrivial in that changes to the game are needed, although a rotational warp is not particularly complicated. The game must expose data for the late-warp such as projection and view. Rollback support in the game server and its integration with late-warp is also important to synchronize player actions with what is shown on-screen.

Late-warp can be a computationally inexpensive solution to the problem of high latency in remote rendering. We have shown the benefits it gives in terms of user performance to first person shooter games. It is likely this also improves quality of experience and may make the difference between some games being playable and not, even competitive esports titles. Other applications

are more broad, covering any remote interaction with video, such as remotely rendered content creation and robotic teleoperation.

An abstract scene was used intentionally in this study to avoid confounding effects from overly detailed rendering and game elements, for example simplified textures and no weapon view model. We have performed some limited informal experiments with more realistic graphics. In future work we intend to further explore late-warp with modern game graphics, networking, and multiplayer interaction, and are hopeful that the results will remain consistent with those in this paper.

REFERENCES

[1] Sudhir Aggarwal, Hemant Banavar, Amit Khandelwal, Sarit Mukherjee, and Sampath Rangarajan. 2004. Accuracy in Dead-Reckoning Based Distributed Multi-Player Games. In *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games* (Portland, Oregon, USA) *(NetGames '04)*. Association for Computing Machinery, New York, NY, USA, 161–165. https://doi.org/10.1145/1016540.1016559

[2] Dmitry Andreev. 2010. Real-Time Frame Rate up-Conversion for Video Games: Or How to Get from 30 to 60 Fps for "Free". In *ACM SIGGRAPH 2010 Talks* (Los Angeles, California) *(SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 16, 1 pages. https://doi.org/10.1145/1837026.1837047

[3] Dean Beeler, Ed Hutchins, and Paul Pedriana. 2016. Asynchronous Spacewarp. https://developer.oculus.com/blog/asynchronous-spacewarp/ Oculus Blog Post.

[4] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. 2004. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003®. In *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games* (Portland, Oregon, USA) *(NetGames '04)*. Association for Computing Machinery, New York, NY, USA, 144–151. https://doi.org/10.1145/1016540.1016556

[5] Yahn W. Bernier. 2003. Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization. (2003).

[6] Kuan-Ta Chen, Polly Huang, and Chin-Laung Lei. 2006. How sensitive are online gamers to network quality? *Commun. ACM* 49 (11 2006), 34–38. https://doi.org/10.1145/1167838.1167859

[7] Shenchang Eric Chen and Lance Williams. 1993. View Interpolation for Image Synthesis. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (Anaheim, CA) *(SIGGRAPH '93)*. Association for Computing Machinery, New York, NY, USA, 279–288. https://doi.org/10.1145/166117.166153

[8] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. 2012. The Brewing Storm in Cloud Gaming: A Measurement Study on Cloud to End-User Latency. In *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games* (Venice, Italy) *(NetGames '12)*. IEEE Press, Article 2, 6 pages.

[9] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. 2014. A Hybrid Edge-Cloud Architecture for Reducing On-Demand Gaming Latency. *Multimedia Systems* 20 (10 2014), 503–519. https://doi.org/10.1007/s00530-014-0367-z

[10] Mark Claypool. 2005. The Effect of Latency on User Performance in Real-Time Strategy Games. *Comput. Netw.* 49, 1 (Sept. 2005), 52–70.

[11] Mark Claypool and Kajal Claypool. 2006. Latency and Player Actions in Online Games. *Commun. ACM* 49, 11 (Nov. 2006), 40–45. https://doi.org/10.1145/1167838.1167860

[12] Mark Claypool and Kajal Claypool. 2010. Latency Can Kill: Precision and Deadline in Online Games. In *Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems* (Phoenix, Arizona, USA) *(MMSys '10)*. Association for Computing Machinery, New York, NY, USA, 215–222. https://doi.org/10.1145/1730836.1730863

[13] Piotr Didyk, Elmar Eisemann, Tobias Ritschel, Karol Myszkowski, and Hans-Peter Seidel. 2010. Perceptually-motivated real-time temporal upsampling of 3D content for high-refresh-rate displays. *Computer Graphics Forum (Proc. of Eurographics)* 29, 2 (2010), 713–722. http://graphics.tudelft.nl/Publications-new/2010/DERMS10c

[14] Sebastian Friston, Tobias Ritschel, and Anthony Steed. 2019. Perceptual Rasterization for Head-Mounted Display Image Synthesis. *ACM Trans. Graph.* 38, 4, Article 97 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3323033

[15] Benjamin Goyette. 2016. Fighting Latency on Call of Duty Black Ops III. (2016). https://www.gdcvault.com/play/1023220/ GDC.

[16] Akimitsu Hogge. 2019. Controller to Display Latency in 'Call of Duty'. (2019). https://www.gdcvault.com/play/1026327/ GDC.

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

[18] Teemu Kämäräinen, Matti Siekkinen, Jukka Eerikäinen, and Antti Ylä-Jääski. 2018. CloudVR: Cloud Accelerated Interactive Mobile Virtual Reality. In *Proceedings of the 26th ACM International Conference on Multimedia* (Seoul, Republic of Korea) *(MM '18)*. Association for Computing Machinery, New York, NY, USA, 1181–1189. https://doi.org/10.1145/3240508.3240620

[19] Babis Koniaris, Maggie Kosek, David Sinclair, and Kenny Mitchell. 2017. Real-Time Rendering with Compressed Animated Light Fields. In *Proceedings of the 43rd Graphics Interface Conference* (Edmonton, Alberta, Canada) *(GI '17)*. Canadian Human-Computer Communications Society, Waterloo, CAN, 33–40.

[20] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, N. Dai, and H. Lee. 2019. Furion: Engineering High-Quality Immersive Virtual Reality on Today's Mobile Devices. *IEEE Transactions on Mobile Computing* (2019), 1–1.

[21] John R. Lange, Peter A. Dinda, and Samuel Rossoff. 2008. Experiences with Client-Based Speculative Remote Display. In *USENIX 2008 Annual Technical Conference* (Boston, Massachusetts) *(ATC'08)*. USENIX Association, USA, 419–432.

[22] Kyungmin Lee, David Chu, Eduardo Cuervo, Johannes Kopf, Yury Degtyarev, Sergey Grizan, Alec Wolman, and Jason Flinn. 2015. Outatime: Using Speculation to Enable Low-Latency Continuous Interaction for Mobile Cloud Gaming. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services* (Florence, Italy) *(MobiSys '15)*. Association for Computing Machinery, New York, NY, USA, 151–165. https://doi.org/10.1145/2742647.2742656

[23] Andrew Lippman. 1980. Movie-Maps: An Application of the Optical Videodisc to Computer Graphics. *SIGGRAPH Comput. Graph.* 14, 3 (July 1980), 32–42. https://doi.org/10.1145/965105.807465

[24] Gerrit Lochmann, Bernhard Reinert, Tobias Ritschel, Stefan Müller, and Hans-Peter Seidel. 2014. Real-time Reflective and Refractive Novel-view Synthesis.. In *VMV*. 9–16.

[25] Michael Long and Carl Gutwin. 2018. Characterizing and Modeling the Effects of Local Latency on Game Performance and Experience. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play* (Melbourne, VIC, Australia) *(CHI PLAY '18)*. Association for Computing Machinery, New York, NY, USA, 285–297. https://doi.org/10.1145/3242671.3242678

[26] I Scott MacKenzie. 2012. *Human-computer interaction: An empirical research perspective*. Newnes.

[27] William R. Mark, Leonard McMillan, and Gary Bishop. 1997. Post-Rendering 3D Warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (Providence, Rhode Island, USA) *(I3D '97)*. Association for Computing Machinery, New York, NY, USA, 7–ff. https://doi.org/10.1145/253284.253292

[28] Joanna Nelius. 2020. PCGamer GeForce Now beats Statdia in our input latency testing. https:https://www.pcgamer.com/geforce-now-beats-stadia-in-our-input-latency-testing/. Accessed: 2020-04-13.

[29] Newzoo. 2020. *Global esports market report*. Vol. April 2020 Update. Newzoo.

[30] Oculus Blog, Facebook 2019. *Introducing ASW 2.0: Better Accuracy, Lower Latency*. Oculus Blog, Facebook. https://www.oculus.com/blog/introducing-asw-2-point-0-better-accuracy-lower-latency/?fb_comment_id=2229988523715492_2230021217045556.

[31] Kjetil Raaen and Andreas Petlund. 2015. How Much Delay is There Really in Current Games?. In *Proceedings of the 6th ACM Multimedia Systems Conference* (Portland, Oregon) *(MMSys '15)*. Association for Computing Machinery, New York, NY, USA, 89–92. https://doi.org/10.1145/2713168.2713188

[32] Bernhard Reinert, Johannes Kopf, Tobias Ritschel, Eduardo Cuervo, David Chu, and Hans-Peter Seidel. 2016. Proxy-guided image-based rendering for mobile devices. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 353–362.

[33] Riot Games. 2020. *Valorant*.

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

[35] Andre Schollmeyer, Simon Schneegans, Stephan Beck, Anthony Steed, and Bernd Froehlich. 2017. Efficient Hybrid Image Warping for High Frame-Rate Stereoscopic. *IEEE Transactions on Visualization and Computer Graphics* PP (01 2017), 1–1. https://doi.org/10.1109/TVCG.2017.2657078

[36] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. 1998. Layered Depth Images. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. Association for Computing Machinery, New York, NY, USA, 231–242. https://doi.org/10.1145/280814.280882

[37] Nathan Sheldon, Eric Girard, Seth Borg, Mark Claypool, and Emmanuel Agu. 2003. The Effect of Latency on User Performance in Warcraft III. In *Proceedings of the 2nd Workshop on Network and System Support for Games* (Redwood City, California) *(NetGames '03)*. Association for Computing Machinery, New York, NY, USA, 3–14. https://doi.org/10.1145/963900.963901

[38] Josef Spjut, Ben Boudaoud, Kamran Binaee, Jonghyun Kim, Alexander Majercik, Morgan McGuire, David Luebke, and Joohwan Kim. 2019. Latency of 30 ms Benefits First Person Targeting Tasks More Than Refresh Rate Above 60 Hz. In *SIGGRAPH Asia 2019 Technical Briefs* (Brisbane, QLD, Australia) *(SA '19)*. Association for Computing Machinery, New York, NY, USA, 110–113. https://doi.org/10.1145/3355088.3365170

[39] Josef Spjut, Ben Boudaoud, Kamran Binaee, Alexander Majercik, Morgan McGuire, and Joohwan Kim. 2019. FirstPersonScience: Quantifying Psychophysics for First Person Shooter Tasks. In *UCI Esports Conference*.

[40] Josef Spjut, Ben Boudaoud, Pyarelal Knowles, Zander Majercik, Morgan McGuire, and Joohwan Kim. 2019. FirstPersonScience. https://github.com/NVlabs/abstract-fps

[41] Art Tevs, Ivo Ihrke, and Hans-Peter Seidel. 2008. Maximum Mipmaps for Fast, Accurate, and Scalable Dynamic Height Field Rendering. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games* (Redwood City, California) *(I3D '08)*. Association for Computing Machinery, New York, NY, USA, 183–190. https://doi.org/10.1145/1342250.1342279

[42] Valve Software. 2012. *Counter-Strike: Global Offensive*.

[43] J. M. P. van Waveren. 2016. The Asynchronous Time Warp for Virtual Reality on Consumer Hardware. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology* (Munich, Germany) *(VRST '16)*. Association for Computing Machinery, New York, NY, USA, 37–46. https://doi.org/10.1145/2993369.2993375