2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA) | 979-8-3503-2658-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISCA59077.2024.00097

GameStreamSR: Enabling Neural-Augmented Game Streaming on Commodity Mobile Platforms

Sandeepa Bhuyan, Ziyu Ying, Mahmut T. Kandemir, Mahanth Gowda and Chita R. Das
Department of Computer Science and Engineering, The Pennsylvania State University, State College, USA
{sxb392, zjy5087, mtk2, mahanth.gowda, cxd12} @psu.edu

***Abstract*—Cloud gaming (also referred to as Game Streaming) is a rapidly emerging application that is changing the way people enjoy video games. However, if the user demands a high-resolution (e.g., 2K or 4K) stream, the game frames require high bandwidth and the stream often suffers from a significant number of frame drops due to network congestion degrading the Quality of Experience (QoE). Recently, the DNN-based Super Resolution (SR) technique has gained prominence as a practical alternative for streaming low-resolution frames and upscaling them at the client for enhanced video quality. However, performing such DNN-based tasks on resource-constrained and battery-operated mobile platforms is very expensive and also fails to meet the real-time requirement (60 frames per second (FPS)). Unlike traditional video streaming, where the frames can be downloaded and buffered, and then upscaled by their playback turn, Game Streaming is real-time and interactive, where the frames are generated on the fly and cannot tolerate high latency/lags for frame upscaling. Thus, state-of-the-art (SOTA) DNN-based SR cannot satisfy the mobile Game Streaming requirements.**

Towards this, we propose *GameStreamSR*, a framework for enabling real-time Super Resolution for Game Streaming applications on mobile platforms. We take visual perception nature into consideration and propose to only apply DNN-based SR to the regions with high visual importance and upscale the remaining regions using traditional solutions such as bilinear interpolation. Especially, we leverage the depth data from the game rendering pipeline to intelligently localize the important regions, called regions of importance (RoI), in the rendered game frames. Our evaluation of ten popular games on commodity mobile platforms shows that our proposal can enable real-time (60 FPS) neurally-augmented SR. Our design achieves a 13× frame rate speedup (and ≈4× Motion-to-Photon latency improvement) for the reference frames and a 1.6× frame rate speedup for the non-reference frames, which translates to, on average 2× FPS performance improvement and 26-33% energy savings over the SOTA DNN-based SR execution, while achieving about 2dB PSNR gain and better perceptual quality than the current SOTA.

***Index Terms*—games, mobile computing, interactive systems, real-time systems, client/server systems, emerging technologies**

I. Introduction

Video games have evolved tremendously since their inception. Advances in computer graphics techniques and graphics hardware have elevated video games from their humble beginnings as a dot on an oscilloscope [12] to highly immersive and realistic experiences of the present day. Currently, with a worldwide user base of approximately 3.2 billion [98], the video game market is a rapidly expanding business, with the mobile gaming revenue accounting for the majority of the global gaming market in 2022 [99]. The advent of 5G era as well as the recently emergent cloud gaming[1] services such as Microsoft xCloud [66], Sony PlayStation Now [83], NVIDIA GeForce Now [71], etc., encompassing a wide collection of games, which offer Games-as-a-Service (GaaS) [69], have made video games more accessible. Especially, cloud gaming services have enabled experiencing graphics-rich desktop and console games on mobiles, which was previously not possible owing to power constraints and resource limitations of mobile SoCs for supporting high-fidelity games.

However, cloud gaming on mobile platforms, demands a high network bandwidth for streaming high-resolution game frames (2K or 4K) and often suffers from significant frame drops even when utilizing a 5G or WiFi network [8]. To mitigate this, the emerging technique of *Super Resolution* (SR), which involves upscaling low-resolution frames to a higher resolution, can be leveraged by streaming low-resolution frames and upscaling them at the client. A significant number of SR techniques from both industry (NVIDIA DLSS [70], Intel XeSS [44], Topaz Video AI [88], etc.) as well as academia ([9], [17], [22], [54], [95]) showcase the **impressive quality improvements** of DNN-based SR models.

Unfortunately, due to their compute-heavy nature, these models are mostly restricted to desktops/servers and lack widespread adoption on mobile platforms. Even with the acceleration techniques [43], [51], [100], [108] or hardware accelerators (TPU and NPU), they are yet to meet 60 FPS frame rate for producing high-resolution frames.

Towards achieving real-time Super Resolution on mobile platforms, NEMO [100] leverages the temporal similarity in video frames by reusing the high-quality super-resoluted reference frame to reconstruct the subsequent non-reference frames, thus amortizing SR processing over a video stream. While this works for video streaming, where the frames are already available and hence can be downloaded and buffered, such a technique is not amenable for Game Streaming owing to its real-time and interactive nature, which requires on the fly game frame generation based on user input. Also, due to the compute-intensive nature of DNN-based SR, this technique incurs a significant lag (> 200 ms) during the reference frame upscaling, thus violating the stringent real-time requirements (16.66 ms) for games, which could even result in "Game over" scenarios in multiplayer games. Observing this, we want

[1]We use the terms "cloud gaming" and "game streaming" interchangeably throughout this paper.

979-8-3503-2658-1/24/$31.00 ©2024 IEEE
DOI 10.1109/ISCA59077.2024.00097

to ask, *what can we do to support* ***high-resolution*** *Game Streaming in* ***real-time*** *on commodity mobile platforms?*

To answer the above question, two important insights can guide us in developing a solution: (1) owing to its interactive nature, the gamer's attention is mostly focused on a specific region of the frame, and (2) a decrease in the size of the input low-resolution frame also decreases the SR generation latency. These insights motivate us to *only apply DNN-based SR to the important region (RoI), instead of the entire frame in [100], while still maintaining a satisfying video quality.* However, achieving this is *not* trivial due to the following **challenges**: First, *how to decide the size of RoI area in order to meet the real-time requirement?* Second, *how to precisely localize the RoI with minimal overheads and maximum quality gains?* Finally, *how to efficiently utilize the captured RoI in order to maintain the quality for the entire frame?*

Towards this, we propose **GameStreamSR**, which, to the best of our knowledge, is the first work towards enabling *real-time* SR on mobile platforms for *high-resolution* Game Streaming. Our major **contributions** are as follows:

- We conduct thorough characterizations for DNN-based SR on commodity mobile platforms and observe the trade-offs between the execution latency and the input resolution, which presents a potential opportunity to achieve real-time SR performance by performing the DNN upscaling on a smaller (but important) frame region (RoI).
- We identify the unique property of the game applications, i.e., "depth data", which reveals the distance between the game objects and the camera, that can be captured during the graphics rendering without any extra overheads. This presents an *energy-efficient* opportunity to determine the RoI for which the DNN-based SR should be applied, compared to the power-hungry camera-based eye-tracking solutions on mobile devices.
- We design two complementary schemes to capture and utilize these two opportunities. First, we propose a depth-guided RoI detection at the server. Then, we utilize the extracted RoI information at the client to upscale the RoI area using DNN-based SR models on the mobile NPU for enhancing quality and simultaneously upscale the remaining region via bilinear interpolation on the mobile GPU in a parallel manner for boosting performance.
- We implemented our design on two state-of-the-art commodity mobile platforms. Evaluations with ten popular gaming workloads demonstrate that our design can support real-time (60 FPS) neurally-enhanced SR by providing 13× (≈4× Motion-to-Photon latency improvement) and 1.6× speedup for the reference and non-reference frames, respectively, compared to the SOTA [100] design. This translates to, on average, ≈2× performance speedup and 26-33% overall energy savings, while providing around 2dB average PSNR as well as better perceptual quality improvement over the SOTA [100]. Our design is hardware, game as well as video codec (unlike SOTA) *agnostic*. This allows utilization of commodity hardware with *no* additional game engine extension and *no* video codec (SW/HW) modification.

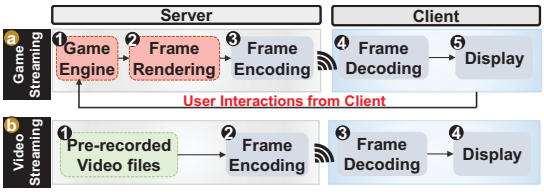


Fig. 1: (a) Game Streaming; (b) Video Streaming pipelines.

II. Background and Motivation

In this section, we first present the end-to-end Game Streaming pipeline and the reasons driving Super Resolution as an attractive solution for Game Streaming. Next, we discuss the limitations of the state-of-the-art (SOTA) SR for video streaming on mobile platforms, which motivates us to further investigate the potential opportunities for achieving real-time SR for Game Streaming on mobile devices.

A. Background

Game Streaming Pipeline: In a typical Game Streaming application (Fig. 1ⓐ), on receiving user input from the client, the game engine (which consists of the game logic) evaluates the next state of the game environment (step-❶) and invokes graphics API calls to the GPU to generate (or render) the next frame (step-❷). The raw rendered frames are then compressed/encoded (step-❸) before transmitting them to the client over the network. Akin to the video streaming pipeline (Fig. 1ⓑ), the client then decodes the received encoded frame (step-❹) and stores it in the frame buffer for the display processor to present it on the client's screen (step-❺). However, in contrast to video streaming, where the raw videos can be pre-encoded offline and stored at the server and reused by several users later, the video game frames cannot be pre-rendered and stored as they mostly depend on user input which varies with different users. This interactive and immersive feature of the Game Streaming application imposes a real-time (16.66 ms or 60 FPS) video game frame generation constraint, unlike conventional video streaming, thus demanding new optimizations for this use case.

High Fidelity Gameplay on Mobile: Game Streaming applications have enabled mobile gamers to enjoy advanced graphics-laden games by streaming *high-resolution* video game frames, thus offering an enhanced and realistic user experience, which was otherwise not possible earlier given the power and resource constraints of the on-device mobile GPUs. However, due to their real-time and interactive nature, the Game Streaming applications are both *bandwidth-heavy and latency-sensitive*, thus demanding a high network bandwidth and a stable and reliable network connection and often tend to fail to meet the QoE requirements for mobile users. A recent study [8] on high-resolution Game Streaming on mobile clients using both live 5G mmWave network and WiFi network shows significant game frame drops during network transmission – around 44% and 90% of frame drops, respectively. This indicates that supporting high-resolution

Game Streaming on mobile clients, while satisfying QoE, is an ongoing challenge. Especially, the wireless 5G channels fundamentally suffer from a bandwidth-latency tradeoff. The high-bandwidth eMBB channel is meant for higher bandwidth but non-latency-sensitive applications, whereas the low-latency URLLC channel is meant for low latency but very low bandwidth-demanding applications [80]. Towards this, the emerging paradigm, *Super Resolution* (also known as *Super Sampling*), which *upscales* the received low-resolution frames to a higher resolution on the client and thereby alleviates the network bandwidth pressure, can offer a practical solution for enjoying such graphics-laden games on mobile platforms.

Super Resolution (SR): Traditionally, frame resolution upscaling was performed using interpolation-based methods such as bilinear [49], bicubic [94], lanczos [28], etc. Recent advancements in deep learning have led to significant accuracy improvement of DNN-based SR models, thus leading to its widespread adoption and becoming a marketable commodity via NVIDIA, AMD and Intel GPUs. NVIDIA DLSS [70] and Intel XeSS [44] leverage their in-house SR models to boost the game frame rate by rendering the games in low resolution and upscaling them to high resolution (4K) using SR models. Although this may seem attractive, due to their heavy compute demands, they are mostly adopted in desktop devices.

Super Resolution on Mobile Platforms: Despite the availability of neural processing units (NPU) or tensor processing units (TPU) on the commodity mobile devices [29], achieving real-time DNN-based SR performance with a high level of accuracy (in this context, high quality) for generating high-resolution video frames on resource-constrained and limited battery life mobile clients still remains an open challenge. For example, a popular Super Resolution model, EDSR [54], can provide only 4.166 FPS on a high-end mobile device [78].

Towards this, a prior work (SOTA) attempts [100] to exploit the temporal dependency in video frames to attain real-time SR for video streaming. The main idea in [100] involves upscaling only the reference frames using a DNN-based SR model. For the non-reference frames, upscales the motion vectors and the residuals using the traditional bilinear interpolation upscaling technique [49] and reuses the upscaled reference frame to construct the upscaled non-reference frames. As a result, the SR execution latency is amortized over a series of video frames, thus yielding a close to real-time upscaling throughput. This approach seems to fit well for video streaming applications, where usually the video frames are downloaded and, buffered in the playback buffer ahead of time and already processed and ready for display during their turn of playback [93]. However, due to the interactive nature of Game Streaming, this technique cannot satisfy the real-time processing requirements of 60 FPS, as will be demonstrated next.

B. What is the Limitation of the SOTA?

To investigate the applicability of SOTA [100] on edge devices, we implemented and tested it on a few group of pictures (GOPs) of game streams using Samsung Galaxy S8 Tab [78] as the mobile client. A GOP is a collection of one reference frame (keyframe) and a few non-reference frames. The GOP size is also referred to as the keyframe interval. Usually, for streaming applications, the keyframe (or reference frame) interval is preset due to several reasons such as optimizing the keyframe seeking time when reconstructing the non-reference frames, for faster keyframe recovery to reduce buffering delays in case a keyframe is lost during transmission, etc. Unlike video streaming, which has a typical keyframe interval of 4 seconds, the recommended keyframe interval for live streaming is 2 seconds [75], and this duration would be further smaller for fast-paced games with quick-changing scenes. As shown in the SR execution timeline plot for 3 consecutive game stream GOPs (Fig. 2), when upscaling a received 720p game stream to 1440p resolution, we observe that the reference frame upscaling latency is too high (where the DNN-based SR is performed), thus *violating* the QoE as the user would experience noticeable lags during gameplay when upscaling the reference frame. Also, this further exacerbates the situation for the Game Streaming applications, where the reference frame (keyframe) frequency is higher than video streaming. Further, for the non-reference frames that use bilinear interpolation upscaling of motion vectors and residuals, the execution latency violates the 16.66 ms (translating to 60 FPS) requirement for Game Streaming applications, which may lead to a "Game over" scenario for fast-paced action games with strict real-time (60 FPS) requirements [13], [85].

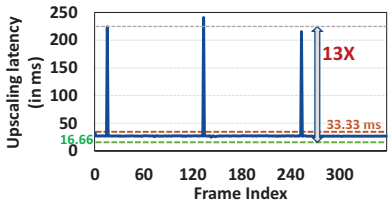
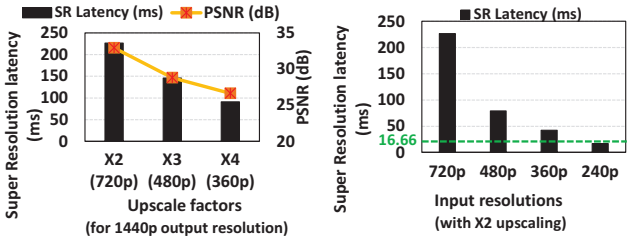


Fig. 2: Super Resolution execution timeline for 3 consecutive GOPs.

C. What are the Potential Opportunities ?

As most commodity high-end mobile devices today support QHD resolution (1440p or 2K) display [31], our upscaled resolution target is 1440p to provide the users with a more visually captivating experience. Note that our primary objective is to reduce the keyframe processing peaks while maintaining quality. So first, to understand the impact on quality and latency when upscaling from different input frame resolutions/sizes to achieve the target resolution (1440p), we analyze the SR execution latencies and quality variation across different upscaling factors. As shown in Fig. 3a, as we increase the upscaling factor, the key observation is that the quality drops significantly, which is unacceptable. Hence, to maintain quality, it is optimal to choose a small upscaling factor. Thus, from a quality-preserving perspective, choosing a 720p (HD) resolution frame as input with an upscaling factor of 2 is a favorable choice. Further, 720p resolution is also the default as well as the most popular streaming resolution used by several streaming platforms [24], [72]. However, the SR latency, when upscaling from 720p input, is too high, owing to the increase in feature map size, thus violating the 16.66 ms deadline.



(a) Different upscaling factors. (b) Different input resolutions.

Fig. 3: Super Resolution execution latency for different: (a) upscale factors and impact on quality; (b) input frame resolutions.

Taking into account the visual perception nature of the human visual system (HVS), especially, in contrast to video streaming which offers more of a relaxing experience, playing games is a very engaging activity, requiring player feedback, thus demanding gamers' attention to be primarily focused on certain areas of the frame during most duration of the gameplay. Driven by this insight and as upscaling an entire 720p resolution frame fails to satisfy the QoE, next we want to investigate if the Super Resolution can be performed on a smaller (important) region of the 720p frame in real-time. Towards this, for a Samsung S8 Tab mobile client, we plot the Super Resolution latency for different input frame resolutions with an upscaling factor of 2 in Fig. 3b and observe that low resolution (e.g., 240p) frame upscaling is able to adhere to the real-time requirements of the Game Streaming application, thus presenting a potential opportunity for leveraging human perception nature for real-time SR of game frames.

Takeaway: *Intelligently* performing the compute-heavy DNN-based SR on smaller areas, preferably on the regions of high visual importance (salient), in a video game frame and applying traditional light-weight interpolation-based SR on the background frame regions can aid in accomplishing real-time Super Resolution for Game Streaming applications.

III. Identifying Key Areas in Game Frames

Driven by the insights from Sec. II-C, with the goal of real-time processing in mind, we first analyze the prior efforts for determining the salient/visually important regions in video frames to understand their feasibility for Game Streaming (Sec. III-A) before identifying an efficient approach for detection of important regions in-game frames (Sec. III-B).

A. Prior Important Region Detection Efforts

Prior saliency detection works can be categorized into two broad types – direct approach and indirect approach.

- *Direct Approaches* involve real-time gaze-tracking on the devices to accurately determine the areas of high visual importance on the frame, based on the user's gaze. In particular, this approach enables what is called *foveated rendering* [38] in the augmented and virtual reality (AR/VR) headsets which are equipped with eye-tracking sensors [63], [65], which reduces the amount of computations by reducing the frame quality in the peripheral vision (outside of foveal-gaze region). Since such energy-efficient sensors are not available on mobile phones, software-based eye-tracking solutions that capture the user's eye movements using the phone's front camera are used as an alternative. However, the camera draws a lot of power, and consequently, its extensive use during the entire gameplay (which is a power-hungry application [8]) can drain the battery even faster. For example, our profiling shows that a Google Pixel 7 Pro device [33] consumes an additional 2.8 watts when using on-device front camera-based eye-tracking during streaming. Further, monitoring user's eye movements using phone camera could potentially lead to privacy concerns. As a result, this approach is not widely adopted in mobile phones for visual saliency detection purposes.
- *Indirect Approaches* involve prediction of visually important regions based on visual cues, such as color, contrast, luminance, texture, etc. in images [15], [107], [111]. Additionally, depth data also provides cues about the spatial information between the objects in an image (especially the foreground and background regions), thus serving as a critical component for saliency detection [20], [50]. Moreover, while DNNs have also been recently employed for saliency detection, they are usually designed for small input frames such as 224×224, 384×384, etc. [10], [42], [77], [96] and such detection on high-resolution frames usually involve either downscaling the input frames to capture the underlying semantic information which results in unavoidable loss of details (and hence, degradation in accuracy) or partitioning the frames into smaller resolutions and batch processing them, thus increasing their processing times.

Owing to the above-mentioned reasons, it is very challenging to attain *real-time salient region detection with good precision*, which is a stringent requirement for high-fidelity Game Streaming applications.

B. How to Identify Important Regions in Real-Time ?

Fortunately, unlike real-world captured images, game frames are computer rendered or generated, and as a part of the graphics rendering process, the depth information, which can facilitate the identification of important regions (as mentioned earlier in Sec. III-A), is generated in the prior (server side) stages of the Game Streaming pipeline. Further, in-game graphics design, in order to improve the rendering performance and enhance the visual quality of 3D game scenes, usually the technique of mipmapping [4] is applied, which basically controls the level of details (LOD) of the game objects based on their distance and size with respect to the camera perspective (i.e., player's viewpoint) in the virtual 3D game environment. This indicates that objects or areas in the game scene that are further away from the player's perspective in the game world have fewer graphics details compared to the nearer ones. Thus, depth of the game object (or the distance of the objects from the player's view) in the game frame controls the amount of details (or LOD) rendered in that frame. Therefore, leveraging depth information in the game rendering stage has the potential to aid in determining

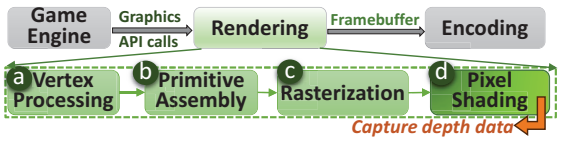


Fig. 4: Game Streaming pipeline on server side.

the region of high visual importance in real-time during the game frame generation.

To understand where to obtain the depth information in the frame rendering stage, we illustrate the components of the game frame rendering/graphics pipeline in Fig. 4. First, when the game engine invokes graphics draw calls to the GPU to render a game frame, the input vertices of the to-be-rendered 3D game objects are processed (ⓐ) and assembled to form geometric primitives (ⓑ) such as points, lines, triangles, etc. which are used to define the shape of the 3D objects. The assembled primitives are then converted (or rasterized) to pixels (ⓒ). Then, in the pixel shading stage (ⓓ), depending on the position of the objects in the game world (using the depth information), the special effects and lighting are applied to each pixel in the frame which decides the final color of each pixel in the game frame. Finally, the color information of each pixel from the pixel shading stage is output into the framebuffer. This suggests that the depth data can be captured from the *depth buffer* (also known as *Z-buffer*). The depth buffer consists of depth value of each pixel in the rendered frame (i.e., the distance of each pixel from the camera/player's perspective plane) and is a 2D array of the same resolution as the rendered frame. Fig. 5 shows a game frame (color components) and its corresponding depth map (grayscale representation of depth) generated from the depth buffer of the frame. The darkness intensity of each pixel in the depth map represents their distance from the player's perspective plane. The nearer regions (foreground) are darker than the further regions (background). This presents us with an **opportunity** to capture the depth information during frame generation with minimal overhead and thus facilitates the *real-time* saliency detection for Game Streaming applications in an *efficient* manner. Towards this, we next propose our design on *how to extract the important regions from the captured depth data* for the game frames at the server and *how to utilize them for intelligent Super Resolution of game frames* on a mobile client.

Fig. 5: (a) A game frame (framebuffer) and (b) its depth map (depth values from depth buffer).

IV. System Design

In this section, we first begin with the high-level overview of our design (Sec. IV-A). We then present our important region (Region of Interest, or RoI) determining technique from a game frame using the depth buffer at the server (Sec. IV-B), followed by our methodology to utilize the detected RoI information to upscale the entire frame at the client (Sec. IV-C).

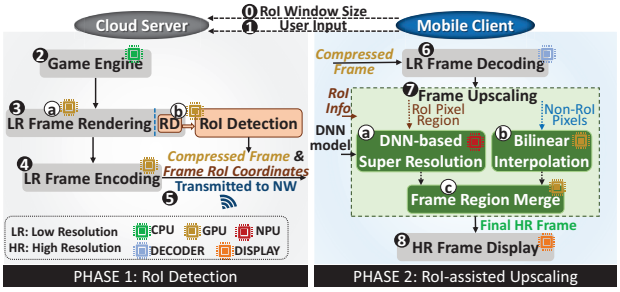


Fig. 6: GameStreamSR design overview.

A. Overview

Fig. 6 illustrates the high-level overview of our proposed framework, **GameStreamSR**, which primarily consists of two phases – *RoI Detection* at the server (Phase-1) and *RoI-assisted Upscaling* at the mobile client (Phase-2).

RoI Detection: As shown in Fig. 6, at the start of the gameplay session, the maximum RoI window size, i.e., the maximum number of pixels in the frame that can be *upscaled in real-time* (16.66 ms) using the DNN-based SR model of the user's choice by the user's mobile device, is determined and conveyed to the server (step-0). Note that, this step is invoked only once for a particular mobile device to determine its maximum compute capability for handling real-time DNN-based SR. Next, during the gameplay, as discussed in Sec. II-A, on reception of the user input from the client, the game engine invokes the graphics API call to the GPU for frame generation/rendering. Based on the maximum real-time processable RoI window size (from step-0), the RoI boundaries can be determined using the depth data during the frame rendering in the GPU (step-3b; more details in Sec. IV-B2). Then, the frame is encoded by the server GPU(step-4). Finally, the encoded frame and its corresponding RoI coordinates are transmitted over the network to the client (step-5).

RoI-assisted Upscaling: At the mobile client, first, the received frame is decoded by the client hardware decoder (step-6). Based on the RoI boundaries, the RoI of the decoded frame is then upscaled using the DNN-based SR model (step-7a) while the remaining areas are upscaled using the bilinear interpolation (traditional non-DNN upscaling) technique (step-7b), followed by the merging of the upscaled RoI and the surrounding regions into the final upscaled framebuffer in step-7c. Finally, the upscaled high-resolution frame can be presented to the player via the display hardware (step-8).

B. Important Region Detection

As our goal is to achieve real-time upscaling of the frames, before delving into details of searching for the RoI in the rendered frame at the server (Sec. IV-B2), the first step is to understand the client device's capability of performing the Super Resolution in real-time (Sec. IV-B1).

(a) Foveal area length. (b) Minimum desired RoI resolution on the low-resolution frame.

Fig. 7: Desired RoI as per human physiology: (a) Foveal visual diameter for normal vision; (b) Desired RoI size on the low-resolution frame.

1) Determining Desired RoI Window Size: We first determine the minimum desired RoI size based on human visual physiology, followed by the maximum RoI size that can be processed in real time by a user's mobile device.

Minimum RoI Window Size: As per the Human Visual System (HVS), the human eyes are very sensitive to the fine visual details around the foveal region (or the center of eye fixation) of the retina, which provides the sharpest vision [26]. Based on this observation, ideally, we need the minimum desired RoI area to be at least as large as the foveal area. This area can be determined using the foveal visual angle and the viewing distance from the eye to the user's mobile device, which are usually fixed. The foveal visual angle (α) for humans is around 5-6° [16]. The typical viewing distance (d) of mobile devices for normal vision is around $30cm$ [106]. Using these, the foveal visual diameter ($2 * d * tan(\alpha/2)$, as shown in Fig. 7a) is around $1.25\ inches$ (= $2*30cm*tan(3°)$ = $3.14cm$) [6]. Since playing games is an engaging activity, the viewing distance may reduce even further, thus reducing the foveal visual diameter. The video frames are processed in units of pixels, thus determining the minimum desired RoI area requires translating the evaluated foveal diameter in terms of the number of pixels. As different mobile devices are designed with different pixel densities, the minimum desired RoI area (pixels) would vary from device to device for the evaluated foveal diameter. For example, for a 2K display resolution Samsung S8 Tab mobile client of pixel density 274 pixels per inch (PPI) [36], 1.25-inch foveal diameter on the 2K display would translate to $\approx$ 343 pixels (=$1.25in*274ppi$). Further, as the RoI detection happens at the server on the rendered low-resolution frame, the minimum desired RoI size on the low-resolution frame would also scale down accordingly. Recall from Sec. II-C that, upscaling by a factor of 2 would guarantee minimal quality loss; thus, upscaling from 720p to 1440p (2K) is an optimal design choice for ensuring better quality. As a result, the minimum desired RoI size on a 720p low-resolution frame for the S8 Tab device would be $\approx$ 172 pixels (=$343/2$ pixels). Therefore, as illustrated in Fig. 7b, the minimum desired RoI size or resolution (foveal resolution) for the low-resolution frames for any mobile client can be generalized as $(pixel\ density * foveal\ visual\ diameter)/scale\ factor$.

Maximum RoI Window Size: Aside the foveal region, which can distinguish the finest details, the central region of the retina surrounding the fovea also accounts for detailed vision to some extent. Beyond the central region, the visual acuity (clarity of vision) drops steeply [92]. To further maximize our quality gains, it is also beneficial to consider the central region as well. However, as increasing the RoI size incurs additional upscaling latency at the client, we need to find the amount of the frame area (in terms of number of pixels) that can be super-resoluted within the real-time constraints by the mobile client. Given the DNN model, such size can be determined by benchmarking the model on the target mobile devices. For example, for running a state-of-the-art DNN-based SR model [54] on the NPU of Samsung S8 Tab in real-time (16.66 ms), the maximum RoI area resolution was found to be 300 pixels resolution. Next, we discuss how the server uses this RoI window size from the client to localize the RoI area in the game frame.

2) RoI Extraction from Depth Buffer: As discussed earlier in Sec. III-B, in computer graphics, the depth information is used to control the level of details in a video game frame. Especially, the near objects (foreground region) are rendered with more details than the far objects (background region) in the game frame. Since loss of details on the near objects will be more noticeable to the player than the far objects, retaining or recovering the visual details in the near objects is imperative. This suggests that the depth data is a prime candidate for determining the RoI in the game frame. Moreover, considering the context of game applications, the depth buffer can be captured during the pixel/fragment shading in the frame rendering stage at the server with *zero* additional overheads. Also, the powerful computing capabilities of the cloud servers can be leveraged to extract the RoI from the depth buffer, thus alleviating any additional processing pressure from the client devices. The depth buffer contains the depth values of each pixel in the frame (i.e., the distance of each pixel from the camera or player's perspective). As illustrated in the depth map (grayscale representation of the depth buffer) of a game frame in Fig. 5, the "darkness intensity" of the pixels represents their "nearness" to the camera or player's perspective plane.

However, extracting the RoI from the depth map may not be a trivial task due to the following challenges: ① The raw depth map can be very detailed and may have varying boundaries between the foreground and background regions across game frames. ② Further, the extracted foreground region may contain objects, which are not the point of focus of the player, e.g., the ground, grass, and walls or buildings at the screen edges, which need to be discarded. ③ Finally, as discussed in Sec. IV-B1, since the extracted RoI bounding box needs to be small enough for real-time upscaling, there is a need to select an RoI window of appropriate size from the filtered foreground. To address these challenges, we can leverage the following insights: ① Games are immersive and engaging activities, where the player explores the virtual game environment and most of the time the player's focus is on the center of the screen [40]; and ② Unlike images, where the RoI of the viewer can vary temporally with shifting focus to different objects across the same image, game videos are fast-motion pictures/frames, thus limiting the chance of the player's focus over multiple RoI objects in the same frame. These insights suggest choosing a *center-biased foreground region* as the player's RoI or visually important region.

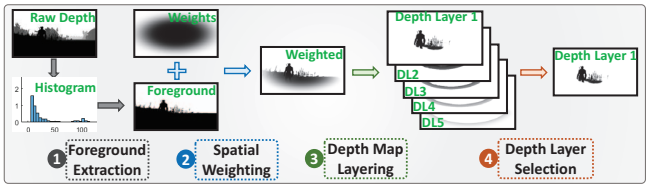


Fig. 8: Depth map pre-processing steps.

Proposed RoI Detection: Driven by these insights, we propose a depth-assisted RoI detection scheme which can be performed in two phases - *Depth map Pre-processing*, followed by *RoI Area Searching* on the processed depth map.

- *Depth Map Pre-processing:* In order to reduce the RoI search complexity and support real-time processing, we first leverage video game characteristics and insight into the player's area of focus to discard extraneous depth information from the raw depth map. Fig. 8 illustrates the steps involved in this pre-processing phase (Foreground Extraction, Spatial Weighting, Depth Map Layering, and Depth Layer Selection) with an example depth map. The first step involves **Foreground Extraction** from the raw depth map (step-❶). The pixels in a depth map correspond to their depth values. Also, the pixels belonging to the same objects have either similar or smoothly varying depth values within a specific range. Hence, we analyze the histogram of the depth map which reveals the distribution of depth values in the depth map to determine the threshold between the foreground and background regions in the depth map. The depth value distribution in the histogram consists of peaks around the objects of similar depths. Based on this, prior works [21], [56] have proposed fine-grained image segmentation methods leveraging the histogram of the depth map. However, since our focus is rather foreground area extraction and, based on our observation that the depth value distributions of the background regions usually have a noticeable gap (or valley) from the foreground area depth values in the histogram, we adopt a coarse-grained approach by identifying such gap in the depth value distribution in the histogram to determine the foreground area depth values in the depth map.

  After this, the second step involves **Spatial Weighting** of each pixel in the depth map based on the likelihood of the player's area of focus in a video game frame (step-❷), followed by distribution of the weighted pixels into different depth layers, which we refer to as **Depth Map Layering** (step-❸). Since a player's gaze tends to be focused mostly on the screen's center (Insight-①), the importance of each pixel in the depth map can be further augmented by adding weights to these pixels based on their screen space position in the video game frame. To achieve this, we use the Gaussian function (normal distribution) [7] to generate a center-biased weight matrix of the same dimension as the depth map and add this weight matrix with the depth map in a pixel-wise manner to highlight the spatial importance of each pixel in the depth map as shown in Fig. 8.

  Next, to narrow down the RoI search space, we segment the spatially-weighted depth map into different depth layers. The depth layers are formed by evenly dividing the depth map into layers based on ranges of weighted depth values. Fig. 8 shows the transformation of the weighted depth map into multiple depth layers. The final pre-processing step involves a single **Depth Layer Selection** from the above depth layers (step-❹). This is done by selecting the depth layer with the maximum overall depth value (summation of depth values in each layer) as shown in Fig. 8. The selected depth layer is then used for performing the RoI search and the remaining layers are discarded.
- *RoI Area Searching:* Using the desired RoI window size from the client (discussed in Sec. IV-B1), the RoI search can then be performed on the processed depth map. In order to reduce the RoI search space, we first perform a coarse-grained search for the RoI with a larger stride to localize the potential RoI position, followed by a fine-grained search with a smaller stride to accurately determine the RoI in the frame. Algo. 1 describes our two-fold RoI search technique.

Algorithm 1: RoI Search on Processed Depth Map.

Input : H, W: Height and width of processed depth map;
h, w: Height and width of RoI search window;
S, s: Coarse-grained and fine-grained RoI search strides;
b: Boundary around the coarse RoI for fine-grained search

1 **for** y = *0 to* H-h *step* S **do** // coarse-grain RoI search
2 **for** x = *0 to* W-w *step* S **do** // in parallel
3 A[x,y] = sum(depth[x:x+h,y:y+w])
4 $[X, Y] = \arg\max_{x \in \{0:S:W-w\},\ y \in \{0:S:H-h\}} A[x, y]$
5 **for** y = Y-b *to* Y+b *step* s **do** // fine-grain RoI search
6 **for** x = X-b *to* X+b *step* s **do** // in parallel
7 B[x,y] = sum(depth[x:x+h,y:y+w])
8 $[X, Y] = \arg\max_{x \in \{0:S:W-w\},\ y \in \{0:S:H-h\}} B[x, y]$

Output: [X,Y]: RoI Coordinates

In the coarse-grained search (line#1-3), the RoI search window of resolution $h \times w$ is traversed using large strides of S pixels apart over the processed depth map of resolution $H \times W$, where h, H, w, W are the heights and widths of the RoI search window and the processed depth map, respectively, such that $S = \max(h, w)/2$. The sum of all the pixels in the depth map falling within the search window during each iteration is stored in an array A (line#3). The pixel position corresponding to the maximum value among the stored sums is chosen as the approximate position of the RoI (line#4). Next, using this approximate RoI position, the fine-grained search for RoI (line#5-7) is performed by traversing the RoI window (of the same dimension $h \times w$ as earlier) with smaller strides (s pixels apart and $s < S$) over a small area of the depth map, including and surrounding the new RoI (i.e., boundary of b pixels surrounding the RoI from the coarse search). Similar to the coarse search process, the sum of all the pixels in the depth map falling within the search window during each iteration of the fine search is stored in an array B (line#7). Finally, the pixel position corresponding to the maximum value among the stored

sums is chosen as our desired RoI position coordinates (line#8). Note that, in case of a tie, if more than one position has the maximum value, as the games are mostly center-biased, the pixel position with the nearest distance to the center of the frame is chosen as the candidate RoI. These RoI coordinates are then conveyed to the client along with the compressed frame for intelligent upscaling of the frames.

Real-time Processing: We want to emphasize that the proposed RoI detection can be done on the server GPU and can leverage parallel GPU cores for most of the operations, thus alleviating the need for back-and-forth *data movement* between GPU and CPU. Further, rendering and encoding low-resolution at the server GPU, compared to rendering and encoding high-resolution frames, would also reduce the GPU utilization (e.g., on an NVIDIA GTX 3080 Ti GPU, the utilization reduces from 79% to 52% when reducing the resolution from 1440p to 720p), thus making room for utilizing the available GPU cores for depth map processing. Moreover, this also helps in reducing the volume of network packet processing both at the server and the client as well as reducing the bandwidth requirement when transmitting low-resolution frames (e.g., bandwidth reduces by 66% when transmitting 720p low-resolution frames along with our RoI information as opposed to transmitting high-resolution 2K frames). Note that Depth Map Processing and RoI Search can be done in GPU (compute) shader cores, which work on multiple pixels in *parallel*, thus making it feasible for real-time detection of RoI from the depth data during the frame rendering stage.

C. Utilizing the Extracted Region

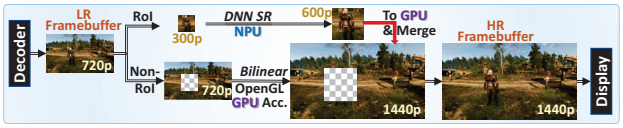


Fig. 9: Frame upscaling at client side.

On receiving a compressed frame and its RoI coordinates pair at the client, the received compressed low-resolution frame is first decoded by the hardware decoder and copied to the LR (low-resolution) framebuffer. Next, the RoI coordinates and desired RoI window size (which is determined at the start of the game session based on client compute capability to support real-time processing, recall from Sec. IV-B1) are used to extract the RoI region pixels from the LR framebuffer which are then sent to the NPU for DNN-based upscaling. Simultaneously, the non-RoI pixels are sent to GPU for hardware-accelerated bilinear interpolation using the OpenGL bilinear filter (GL_LINEAR) [47]. The upscaled RoI region from the NPU is then sent to the GPU and copied into the corresponding pixel positions to merge with the upscaled non-RoI region. Finally, the merged high-resolution frame is deposited on the HR (high-resolution) framebuffer, from where the display hardware reads and displays the frame.

For example, as shown in Fig. 9, after receiving and decoding a 720p low-resolution frame, for upscaling the input frame by a factor of 2 (to 1440p), an S8 tab client extracts and sends the RoI region of size 300x300 (desired RoI window size) from the LR framebuffer using the received RoI coordinates to the NPU and sends the non-RoI pixels to the GPU for upscaling. Note that, these upscaling steps at the NPU and GPU are performed in *parallel*, with the RoI upscaling on NPU taking about 16.2ms, while the non-RoI upscaling on GPU taking only 1.4ms. Finally, the upscaled RoI of size 600x600 is *merged with* the upscaled non-RoI region before copying the final 1440p upscaled frame into the HR framebuffer.

TABLE I: Game Workloads

ID	Game	Genre
G1	Metro Exodus [82]	First Person Shooter
G2	Far Cry 5 [25]	Third Person Shooter
G3	Witcher 3 [76]	Role playing
G4	Red Dead Redemption 2 [74]	Action
G5	Grand Theft Auto V [55]	Adventure
G6	God of War [79]	Action-adventure
G7	Shadow of the Tomb Raider [84]	Survival
G8	A Plague Tale: Requiem [5]	Stealth
G9	Farming Simulator 22 [30]	Simulation
G10	Forza Horizon 5 [73]	Racing

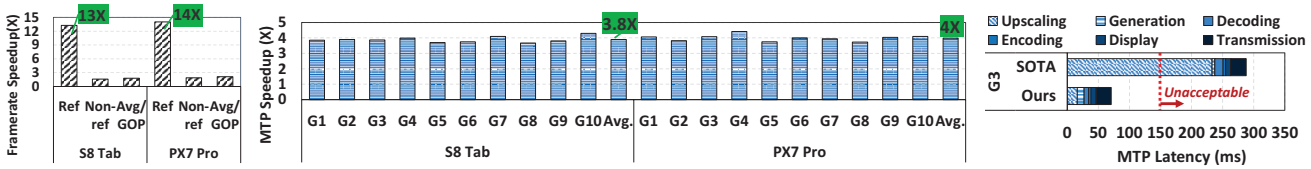
V. Evaluation

In this section, we present our evaluation platforms, workloads, and design configurations and experimental results.

A. Methodology

Platform: Owing to the proprietary and closed-source nature of the popular cloud gaming services (e.g., NVIDIA GeForce Now [71], Microsoft xCloud [66], etc.), we use a powerful gaming desktop workstation equipped with AMD Ryzen 9 5900X desktop 12-core processor, 32GB DDR4 memory, and NVIDIA GeForce GTX 3080 Ti graphics card, as our server setup. For the client side, we use two commodity state-of-the-art mobile/handheld devices from two leading vendors – i) Google Pixel 7 Pro phone equipped with the Google Tensor G2 SoC and edge TPU and ii) Samsung Galaxy S8 tablet equipped with the Qualcomm Snapdragon 8 Gen 1 SoC and hexagon tensor processor. The network communication between the server and client is handled using a high-speed WiFi connection.

Software Setup: At the server side, for retrieving and processing the depth data, we use an open-source tool called ReShade [19], which captures the graphics API calls (OpenGL, direct3D, and Vulkan) and exposes an API similar to HLSL (High-Level Shading Language) [64], which allows the developers to execute shader operations on color and depth data. Game streaming is handled by two open-source applications: Sunshine [59] for the server and Moonlight [37] for the client, implementing NVIDIA's GameStream [18] host and client protocol, respectively. At the client side, we use the TensorFlow Lite NNAPI delegate [34] for using the NPU/TPU on the mobile device to perform the DNN-based SR. We use the TensorFlow Lite benchmark tool [35] to measure



(a) Speedup w.r.t. SOTA. (b) (End-to-end / Motion-to-Photon) MTP performance speedup w.r.t. SOTA. (c) MTP latency breakdown for G3.

Fig. 10: Performance Results: (a) Speedup; (b) MTP latency improvement for reference frame; (c) MTP latency breakdown of G3 for reference frame.

the Super Resolution performance. As the current smartphone batteries are tightly integrated into the phone's body (making it infeasible for hardware power measurement), we use the Termux application [27] to collect the voltage and current readings from the Android kernel and calculate the power.

To demonstrate the effectiveness of our proposal, we use a popular Super Resolution DNN, EDSR [54], with 16 residual blocks and 64 channels and an upscale factor of 2, and use ten widely popular high graphics-fidelity game benchmarks [68] listed in Table I across different game genres as our dataset representing varied game scenes or surroundings.

Baseline (SOTA): We implement the state-of-the-art (SOTA) Super Resolution on mobile devices for streaming application, NEMO [100] as our baseline. All components of the NEMO pipeline were ported to our setup. Note that, NEMO requires video codec internal modifications to upscale the non-reference frames from the upscaled reference frame. Hence, it fails to use the energy-efficient mobile hardware video decoder, and instead relies on libvpx decoder [32], software implementation of VP9 codec, on the ARM Neon mobile CPU.

B. Results

We evaluate our proposed design across four metrics critical for smooth gameplay on a mobile client, i.e., frame rate, motion-to-photon latency, energy consumption, and visual quality, and present our improvements over the SOTA (baseline) for each game on our mobile platforms as follows:

Frame rate: We observe that the output frame rate (frames per second) at the client is primarily influenced by the upscaling performance, as this stage is executed towards the end of the frame display in the game streaming pipeline. As shown in Fig. 10a, compared to the SOTA, our optimization achieves about 13× and 14× upscaling performance speedup for the reference frame on the S8 tab and Pixel 7 Pro phone, respectively. This in turn leads to the output frame rate improving from 4.6 FPS to 61.7 FPS on the S8 tab and 4.3 FPS to 61 FPS on the Pixel phone for the reference frame, thus minimizing the lags/frame stutters observed due to the super-resolution processing (as demonstrated earlier in Sec. II-B and Fig. 2) and hence, enables real-time processing (60 FPS). This speedup mainly results from our design choice of DNN-based upscaling of the RoI on the NPU and offloading the frame's remaining regions to the GPU for hardware-accelerated upscaling via bilinear interpolation technique. In contrast, the SOTA [100] implements DNN-based upscaling for the entire reference frame. Moreover, our design also achieves more than 1.5× upscaling performance speedup for the non-reference frames on both mobile clients. This is because the SOTA design involves additional upscaling of the motion vectors and the residuals of the non-reference frames (bilinear interpolation on the CPU), which can be avoided by using our approach.

The performance boosts of both reference and non-reference frames further translate to around 2× upscaling speedup for a group of pictures (GOP, which is a sequence of a reference frame followed by its dependent 59 non-reference frames) across the mobile platforms. Note that, the upscaling performance speedup (and ultimately, the output frame rate) across the games did not show any significant difference and thus, was not included in the speedup plot.

Motion-to-Photon Latency: Another important requirement for gaming applications is the Motion-to-Photon (MTP) latency, the time between a player's action to the resulting frame being displayed on the screen, which impacts the player's performance. Cloud gaming applications can tolerate MTP latency up to 150 ms [86], but this threshold can be as low as 100 ms for fast-paced game genres (shooter, action, racing, etc.) [52]. As shown in Fig. 10b, compared to the SOTA, our optimization achieves around 3.8× and 4× end-to-end MTP latency improvement for the reference frame on the S8 tab and Pixel 7 Pro phone, respectively. This improvement can be understood by observing the MTP latency breakdown (Fig. 10c) across the Game Streaming pipeline stages for the Witcher 3 (G3) game when played on a Pixel 7 Pro phone. As observed, our upscaling technique requires only 16.4 ms for the reference frame processing. In contrast, the upscaling stage in SOTA (∼233 ms) itself violates the MTP requirement. This is problematic for video games as the reference frames occur much more frequently in game scenes compared to traditional video streaming applications. Further, both our technique and the SOTA achieve less than 100 ms latency for the non-reference frames across the gaming workloads. And, finally, our optimization achieves an MTP latency below 70 ms for both reference and non-reference frames across the entire game streaming pipeline and is conformant with MTP latency requirement for all tested games.

Energy: As battery life is a critical concern for mobile clients, we plot the overall energy savings of our design with respect to the SOTA (Fig. 11) across all our gaming workloads. We observe, on an average, 26% and 33% energy savings on the S8 tab and Pixel 7 Pro phone, respectively. The S8 tablet incurs additional energy consumption owing to its larger display size compared to the Pixel device. To understand where the energy savings come from, we present the energy consumption breakdown of our implementation when playing the Witcher

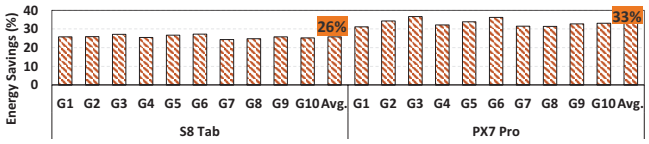


Fig. 11: Overall energy savings w.r.t SOTA.

3 game [76] on the Pixel phone in Fig. 12. Especially, we observe significant energy savings in the decoding stage in our design (reduced from 46% of overall energy consumption in the SOTA to 6% in our design). This saving comes from our approach's flexibility to leverage the energy-efficient hardware decoders on the device, as opposed to the SOTA, which requires using a software implementation of the decoder on the CPU. As the volume of the network packets received and the final high-resolution framebuffer to be displayed are the same in both the designs, our display and network processing energies do not vary compared to the SOTA. Despite upscaling the frame RoI on the NPU and the remaining frame region using hardware-accelerated bilinear interpolation on the GPU for all the frames (both reference and non-reference), our upscaling energy is slightly higher than that of the SOTA (where the upscaling energy constitutes the entire reference upscaling on the NPU and the motion vectors and the residuals upscaling for the non-reference frames on the CPU). Further, we would like to emphasize that, with similar upscaling energy consumption, our design can provide real-time SR for all the frames, which is a stringent necessity for gaming.

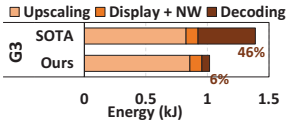


Fig. 12: Energy consumption breakdown.

Quality: We compare our video quality gains against the SOTA using both pixel-wise and perceptual quality assessment approaches. For the *pixel-wise* comparison, we use peak signal-to-noise ratio (PSNR), an objective full-reference metric, to quantify the overall quality of the upscaled frame with respect to the original high-resolution frame. As shown in Fig. 14a, our design achieves a PSNR gain of ≈2dB, on average, across our workloads over SOTA. These quality gains can be understood by zooming into the PSNR variation of Witcher 3 game over consecutive GOPs (shown in Fig. 13). Typically, a PSNR value of 30dB or higher is acceptable for video frames [81], with higher values indicating better frame quality. As shown in Fig. 13, the SOTA (orange line) observes higher PSNR values when upscaling the reference frames using DNN-based SR, but its PSNR values soon fall below 30dB due to accumulation of pixel errors owing to the bilinear interpolation of the subsequent non-reference frames. Although, with our proposal (black line), the overall frame PSNR is slightly lower than the SOTA for the reference frame as we opportunistically use DNN-based upscaling only for the RoI, our PSNR is still consistently above 30dB.

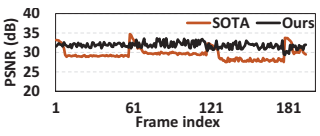
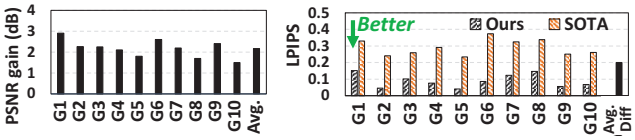


Fig. 13: Transient PSNR variation snapshot for G3 w.r.t SOTA.



(a) PSNR gain w.r.t. SOTA. (b) LPIPS improvement w.r.t. SOTA.

Fig. 14: Quality: (a) Objective comparison w.r.t SOTA, and (b) Subjective perceptual quality comparison. Lower is better.

Next, for the *perceptual* comparison, we use Learned Perceptual Image Patch Similarity (LPIPS) [53], a DNN-based perceptual full-reference metric, which is widely adopted as a standard perceptual similarity metric in computer graphics and vision domains [46], [57], [60]–[62], [91], [97]. LPIPS, ranging from 0 to 1, evaluates the distance between frame regions based on deep features and has shown strong correlation with human perception than the traditional metrics [112]. A lower LPIPS score means that two patches are more perceptually similar. Fig. 14b presents the overall LPIPS score of the upscaled frame with respect to the original high-resolution frame comparing our optimization w.r.t. SOTA. As observed, our design achieves lower LPIPS value than SOTA across all games indicating more perceptual similarity to the original high-resolution frame than SOTA. Also, on average, our LPIPS difference with SOTA is ∼0.2 indicating perceptible improvement over SOTA, as a LPIPS difference of at least 0.15 is typically observed to be visibly discernible [41]. This perceptible quality loss in SOTA is incurred due to the successive bilinear interpolation of the non-reference frames.

To summarize, our design delivers real-time frame throughput (60 FPS) with a MTP latency below 70 ms for all frames and also saves 26-33% energy on mobile devices, while offering better perceptual quality than the SOTA.

VI. Discussion and Future Work

Game Support: At the server side, as we leverage depth data from the generic graphics rendering pipeline, our design can support most 3D games without the cumbersome and intrusive need for modifications in the game engine (which are mostly proprietary). Especially, owing to underlying 3D rendering process similarity with VR games, our design can also extend to Cloud VR gaming. While integrated eye-tracking sensors within VR headsets can offer RoI-detection at client for Cloud VR gaming, these capabilities are often restricted to premium headsets, limiting their accessibility. Our design extends benefits to devices without such advanced sensors, offering a broader and inclusive scope.

However, note that, our depth-guided RoI detection will not support games with limited/constrained game environment perspectives like top-down, flat, or side-scrolling views, which typically lack depth data crucial for RoI-detection. 2D puzzle and platformer games with flat, or side-scrolling views usually

do not complex 3D scenes, and due to their rendering simplicity, they can be directly rendered on the mobile GPU [45]. And, strategy games with top-down perspective view, pose challenges in foreground-background distinguish due to the consistent distance from virtual camera across all frame regions. Further, strategy games are competitive and are usually played using a lot of keyboard shortcuts. Due to lack of enough physical keys on mobile devices, playing such games on mobile devices will be unwieldy and ultimately unenjoyable. Further, supporting RoI-detection in limited-perspective games would require cues from game engines, which are often proprietary and inaccessible. Considering these aspects, the focus of this paper is on streaming broad spectrum of high graphics-fidelity 3D games that utilize first-person (FPP) and third-person perspective (TPP) view in the game environment, and thus would require DNN-based upscaling for a more photo-realistic experience.

Codec Agnostic: Unlike SOTA, our design is video codec agnostic (requiring no decoder modification), and can be readily adapted for existing commodity mobile devices with the popular codec standards like H.265, H.264, etc. Moreover, in contrast to software codec implementation in SOTA, our design leverages the energy-efficient hardware decoders and can potentially achieve further energy savings with minimal hardware additions, as discussed next.

Architectural Insights: Although our design provides 26-33% energy saving over SOTA (Fig. 12), our upscaling energy is still high (85% of the overall energy consumption). This is because the DNN-based upscaling for RoI is very energy-draining. Such observation motivates us to exploit the *temporal redundancy* in the game frames in our RoI-based approach and, thereby, skip performing DNN-based SR for every frame, but with careful design considerations for *minimal quality loss* which has been missed by SOTA. To achieve this, we plan to design an energy-efficient RoI-guided SR-integrated decoder hardware, which not only facilitates *fine-grained* quality control (instead of applying uniform quality across the entire frame like SOTA) and thereby, yields better QoE, but also offers superior energy efficiency compared to the software upscaling implementation employed in SOTA. Fig. 15 depicts the prototype of our design, where the grey boxes represent the generic components in the mobile video decoder hardware, the green boxes show our current RoI-guided upscaling implementation (this work), and the blue boxes represent the additional hardware augmentation (future work). The key idea is to first use our RoI-based upscale engine for upscaling the decoded reference frame (step-❶), cache it in the decoder buffer (step-❷), and then use our SR-integrated hardware decoder to upscale the dependent non-reference frames. To efficiently minimize quality loss in the upscaled non-reference frames, RoI-guided hints can be used in the residual and motion vector (MV) interpolation module (step-❸) for their respective upscaling. For example, using better quality-preserving implementations such as bicubic- or lanczos-based interpolation techniques to upscale the RoI regions while bilinear technique for non-RoI frame regions.

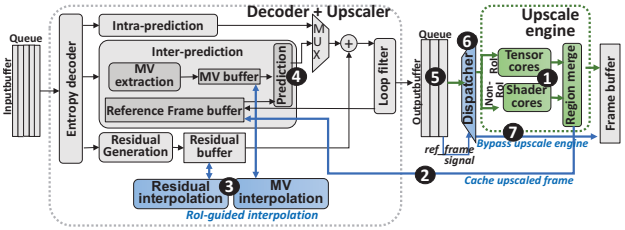


Fig. 15: Proposed RoI-guided SR-integrated video decoder.

The decoder can then use the cached upscaled reference frame and upscaled MV and residuals to reconstruct the non-reference frames (step-❹). Next, when the decoded frames are submitted into the Outputbuffer queue (step-❺), they can be fed into a lightweight demultiplexer called frame dispatcher (step-❻), which can route the frames either to the upscale engine (for reference frames) or directly to the frame buffer (for non-reference frame to *bypass* the upscale engine) (step-❼) depending on the frame type (by probing the frame metadata). Note that, with this prototype, by extending the existing decoder hardware and reusing its components with minimal hardware additions, we expect to save even more energy (as high as 50%), and thus further promote widespread adoption of real-time SR for mobile Game Streaming.

VII. Related Work

Video Processing Optimizations: Prior works [14], [23], [89], [90], [103], [115] have used visual perception-guided schemes to optimize AR/VR video streaming. For example, [14], [23] propose to reduce the headset's display power by modulating pixel colors in frames, [90], [115] propose optimizations to reduce memory traffic and/or energy consumption in AR/VR systems by leveraging human eye sensitivity to color, etc., while [89] proposes a compiler framework for edge to support memory- and power-efficient frame processing. Such approaches can help Game Streaming in later pipeline stages. However, they do not address the main bottleneck in Game Streaming which occurs in earlier stage (i.e, frame drops at network), and ultimately propagates in the pipeline resulting in QoE degradation. Towards this, our real-time SR enablement approach for high-resolution gameplay serves as a practical solution. Another video streaming optimization has been exploiting frame similarities. For example, [105], [109], [110] propose to utilize content similarities to reduce the memory pressure, while [104], [113], [114] aims to reduce the computations at the edge devices for volumetric video streaming (e.g., high-resolution 360°-video, point cloud, etc.). In contrast, mobile Game Streaming is much more challenging than video streaming as discussed earlier.

Super Resolution Acceleration: DNN-based Super Resolution (alias "neural enhancement") has attracted attention from both academia [48], [95], [101], [102] and industry [1], [2], [11], [44], [70] as an effective technique for enhancing the video stream quality. Specifically, LiveNAS [48] and NeuroScaler [102] focus on neural-enhanced live video streaming.

However, they primarily target desktop-level (or even more powerful) platforms and are *not* suitable for mobile devices (the target platform of our work). To facilitate the deployment of DNN-based SR on such resource-constraint devices, various acceleration techniques [43], [51], [100], [108] have been proposed. For example, NEMO [100] proposes to apply the neural SR to a few anchor/reference frames, while MobiSR [51] offloads the such DNN-based SR tasks across CPU, GPU, and DSPs. However, they are yet to achieve real-time performance. In contrast, [43] presents some efficient Super Resolution DNN architectures for mobile/IoT devices, while [108] improves the SR performance via model pruning. However, these techniques are designed for lower input resolutions (e.g., 360p) and *cannot* meet real-time requirements for Game Streaming using a 720p input. Unlike these prior works, which employ their compute to improve the quality evenly across all regions, we focus the quality gains from the SR model only on the *visually important areas* in the frame. By doing so, we are able to achieve QoS-conformant and energy-efficient "real-time performance" for Game Streaming applications.

RoI Detection in Games: Game streaming faces the significant challenge of high bandwidth requirements. Towards this, RoI-based optimizations have been proposed to encode only the important regions (RoIs) at high quality. Existing RoI detection methods are game-specific and require either training game-specific DNN models (like visual attention [3], object detection [67], etc.) lacking generality or manual tagging of game objects [39]. As most popular games are proprietary restricting access to game objects data, this necessitates either waiting for game developers to modify their games to tag game objects and expose this RoI data to users or supporting games that already have this feature. Further, game developers may not prioritize incorporating RoI detection into their game design, as they focus on other aspects of gameplay. Besides, for popular games lacking active developer support, relying on manual tagging by developers may not be feasible. This demands the need for a more *generalized* approach proposed by our design for RoI detection using the depth data more freely accessible to users. Moreover, while [58] relies on depth for RoI-based encoding, its subsequent complex encoding/decoding process to use those depth cues limits its performance (to <30 FPS) falling short of 60 FPS, a necessary requirement for gaming. Finally, [87] mounts the eye and head tracking sensors on the PC monitor to capture RoIs in games. However, it only supports limited games and is impractical for mobile devices due to the need for physical sensor attachment.

VIII. Conclusion

GameStreamSR is the first Game Streaming framework that enables real-time Super Resolution of video game frames on commodity mobile platforms. To achieve this, it exploits the visual perception nature and an inherent feature of the gaming application to capture the depth-guided RoI information from the server and use it at the client to intelligently *upscale* the RoI using DNN-based SR, while the remaining region is upscaled using fast bilinear interpolation. On evaluation using ten popular gaming benchmarks on two commodity mobile platforms, GameStreamSR achieves 13× and 1.6× speedups for the reference and non-reference frames, respectively, with 26-33% energy savings, while providing, on average, 2dB PSNR and better perceptual quality gain over the SOTA.

IX. Acknowledgment

We thank the anonymous reviewers, Anoosh Reddy, Rishabh Jain and other HPCL members for their feedback on this paper. This research is supported in part by NSF grants #2211018, #1931531, and #2116962.

References

[1] Advanced Micro Devices, Inc., "AMD FidelityFX - Super Resolution 2 (FSR 2)," https://gpuopen.com/fidelityfx-superresolution-2/, 2023.
[2] Advanced Micro Devices, Inc, "AMD Virtual Super Resolution," https://www.amd.com/en/technologies/vsr, 2023.
[3] H. Ahmadi, S. Zadtootaghaj, F. Pakdaman, M. R. Hashemi, and S. Shirmohammadi, "A skill-based visual attention model for cloud gaming," *IEEE Access*, vol. 9, pp. 12 332–12 347, 2021.
[4] Arm, "Real-time 3D Art Best Practices - Texturing," https://developer.arm.com/documentation/102449/0100/Mipmapping, 2023.
[5] Asobo Studio SAS, "A Plague Tale Requiem," "https://www.focus-entmt.com/en/games/a-plague-tale-requiem", 2023.
[6] B. Bauer, "Visual Angle Primer," http://www.gta.igs.net/~bgbauer/trentweb/366W04/images/visualangle.html, 2023.
[7] M. Bensimhoun, "N-dimensional cumulative function, and other useful facts about gaussians and normal densities," *Jerusalem, Israel, Tech. Rep*, pp. 1–8, 2009.
[8] S. Bhuyan, S. Zhao, Z. Ying, M. T. Kandemir, and C. R. Das, "End-to-end characterization of game streaming applications on mobile platforms," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 6, no. 1, pp. 1–25, 2022.
[9] bloc97, "Anime4K," https://github.com/bloc97/Anime4K, 2023.
[10] G. Boesch, "VGG Very Deep Convolutional Networks (VGGNet) – What you need to know," https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/, 2023.
[11] Brian Choi, "Pixel Perfect: RTX Video Super Resolution Now Available for GeForce RTX 40 and 30 Series GPUs," https://blogs.nvidia.com/blog/2023/02/28/rtx-video-super-resolution/, 2023.
[12] Brookhaven National Laboratory, "The First Video Game?" https://www.bnl.gov/about/history/firstvideo.php, 2022.
[13] Build-Gaming-Computers.com, "What's the Best Frame Rate for Gaming ?" https://www.build-gaming-computers.com/best-frame-rate-for-pc-gaming.html, 2021.
[14] K. Chen, B. Duinkharjav, N. Ujjainkar, E. Shahan, A. Tyagi, J. He, Y. Zhu, and Q. Sun, "Imperceptible Color Modulation for Power Saving in VR/AR," 2023.
[15] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu, "Global contrast based salient region detection," *IEEE transactions on pattern analysis and machine intelligence*, pp. 569–582, 2014.
[16] J. A. Choi, J.-S. Kim, H.-Y. L. Park, H. Park, and C. K. Park, "The foveal position relative to the optic disc and the retinal nerve fiber layer thickness profile in myopia," *Investigative ophthalmology & visual science*, pp. 1419–1426, 2014.
[17] M. V. Conde, U.-J. Choi, M. Burchi, and R. Timofte, "Swin2sr: Swinv2 transformer for compressed image super-resolution and restoration," in *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*. Springer, 2023, pp. 669–687.
[18] N. Corporation, "NVIDIA GameStream," "https://www.nvidia.com/en-us/support/gamestream/", 2023.
[19] crosire, "Reshade," "https://reshade.me", 2023.
[20] K. Desingh, K. M. Krishna, D. Rajan, and C. Jawahar, "Depth really matters: Improving visual salient region detection with depth." in *BMVC*, 2013, pp. 1–11.
[21] T. H. Dinh, M. T. Pham, M. D. Phung, D. M. Nguyen, Q. V. Tran *et al.*, "Image Segmentation Based on Histogram of Depth and an Application in Driver Distraction Detection," in *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*. IEEE, 2014, pp. 969–974.

[22] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[23] Duinkharjav, Budmonde and Chen, Kenneth and Tyagi, Abhishek and He, Jiayi and Zhu, Yuhao and Sun, Qi, "Color-Perception-Guided Display Power Reduction for Virtual Reality," *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, p. 144:1–144:16, 2022.

[24] Emily Krings, "Decoding 720p: The Best Streaming Resolution Settings for Live Quality Video," https://www.dacast.com/blog/best-h-264-encoder-settings-for-live-streaming/, 2022.

[25] U. Entertainment, "Far Cry 5," "https://www.ubisoft.com/en-us/game/far-cry/far-cry-5", 2023.

[26] Everyday Sight, "Fovea," https://www.everydaysight.com/fovea/, 2023.

[27] F. Fornwall, "Termux," "https://termux.com/", 2023.

[28] Francesco Mazzoli, "Lánczos interpolation explained," urlhttps://mazzo.li/posts/lanczos.html, 2022.

[29] A. Frumusanu, "Google's IP: Tensor TPU/NPU," https://tinyurl.com/4vbzxh5n, 2021.

[30] GIANTS Software, "Farming Simulator 22," "https://www.farming-simulator.com/", 2023.

[31] Gines, "All The Best Phones With 2K Display Listed | 2023 Edition," https://tinyurl.com/bd7aduc3, 2023.

[32] Google LLC, "libvpx," https://chromium.googlesource.com/webm/libvpx, 2023.

[33] ——, "Pixel 7 Pro," https://store.google.com/product/pixel_7_pro?hl=en-US, 2023.

[34] ——, "TensorFlow Lite NNAPI," "https://www.tensorflow.org/lite/android/delegates/nnapi", 2023.

[35] ——, "TensorFlow Performance Measurement," "https://www.tensorflow.org/lite/performance/measurement", 2023.

[36] GSMArena.com, "Samsung Galaxy Tab S8," https://www.gsmarena.com/samsung_galaxy_tab_s8-11343.php, 2023.

[37] C. Gutman, D. Waxemberg, A. Neyer, M. Bergeron, A. Hennessy, and A. Campbell, "Moonlight," "https://moonlight-stream.org/", 2023.

[38] D. Heaney, "Here's The Exact Performance Benefit Of Foveated Rendering On Quest Pro," https://uploadvr.com/quest-pro-foveated-rendering-performance/, 2022.

[39] M. Hegazy, K. Diab, M. Saeedi, B. Ivanovic, I. Amer, Y. Liu, G. Sines, and M. Hefeeda, "Content-aware Video Encoding for Cloud Gaming," in *Proceedings of the 10th ACM multimedia systems conference*, 2019, pp. 60–73.

[40] S. K. Holm, T. Häikiö, K. Olli, and J. K. Kaakinen, "Eye movements during dynamic scene viewing are affected by visual attention skills and events of the scene: evidence from first-person shooter gameplay videos," *Journal of Eye Movement Research*, 2021.

[41] Q. Hou, A. Ghildyal, and F. Liu, "A perceptual quality metric for video frame interpolation," in *European Conference on Computer Vision*. Springer, 2022, pp. 234–253.

[42] Hugging Face, "Vision Transformer (large-sized model) ," https://huggingface.co/google/vit-large-patch32-384, 2023.

[43] A. Ignatov, R. Timofte, M. Denna, A. Younes, G. Gankhuyag, J. Huh, M. K. Kim, K. Yoon, H.-C. Moon, S. Lee *et al.*, "Efficient and accurate quantized image super-resolution on mobile npus, mobile ai & aim 2022 challenge: report," in *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*. Springer, 2023, pp. 92–129.

[44] Intel Corporation, "XeSS Super Sampling," https://www.intel.com/content/www/us/en/products/docs/arc-discrete-graphics/xess.html, 2023.

[45] Jacqueline Zalace, "10 Best Platformers On iOS And Android," https://shorturl.at/cgrsO, 2023.

[46] J. Jiang, C. Wang, X. Liu, and J. Ma, "Deep learning-based face super-resolution: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–36, 2021.

[47] Kevin, "Android Lesson Six: An Introduction to Texture Filtering," https://tinyurl.com/3y7t3b7n, 2012.

[48] J. Kim, Y. Jung, H. Yeo, J. Ye, and D. Han, "Neural-enhanced live streaming: Improving live video ingest via online learning," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2020, p. 107–125.

[49] E. J. Kirkland and E. J. Kirkland, "Bilinear interpolation," *Advanced Computing in Electron Microscopy*, pp. 261–263, 2010.

[50] C. Lang, T. V. Nguyen, H. Katti, K. Yadati, M. Kankanhalli, and S. Yan, "Depth matters: Influence of depth cues on visual saliency," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part II 12*. Springer, 2012, pp. 101–115.

[51] R. Lee, S. I. Venieris, L. Dudziak, S. Bhattacharya, and N. D. Lane, "Mobisr: Efficient on-device super-resolution through heterogeneous mobile processors," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019.

[52] Z. Li, H. Melvin, R. Bruzgiene, P. Pocta, L. Skorin-Kapov, and A. Zgank, "Lag compensation for first-person shooter games in cloud gaming," in *Autonomous Control for a Reliable Internet of Services: Methods, Models, Approaches, Techniques, Algorithms, and Tools*. Springer International Publishing Cham, 2018, pp. 104–127.

[53] Lightning AI Inc., "Learned Perceptual Image Patch Similarity (LPIPS)," https://torchmetrics.readthedocs.io/en/stable/image/learned_perceptual_image_patch_similarity.html, 2023.

[54] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 136–144.

[55] R. N. Limited, "Grand Theft Auto V," "https://www.rockstargames.com/gta-v", 2023.

[56] C. Lin, Y. Zhao, J. Xiao, and T. Tillo, "Depth Map Coding Using Histogram-Based Segmentation and Depth Range Updating," *KSII Transactions on Internet and Information Systems (TIIS)*, pp. 1121–1139, 2015.

[57] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 651–15 663, 2020.

[58] Y. Liu, S. Dey, and Y. Lu, "Enhancing video encoding for cloud gaming using rendering information," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 1960–1974, 2015.

[59] LizardByte, "Sunshine," "https://docs.lizardbyte.dev/projects/sunshine/en/latest/about/overview.html", 2023.

[60] A. Lugmayr, M. Danelljan, R. Timofte, M. Fritsche, S. Gu, K. Purohit, P. Kandula, M. Suin, A. Rajagoapalan, N. H. Joon *et al.*, "Aim 2019 challenge on real-world image super-resolution: Methods and results," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 2019, pp. 3575–3583.

[61] C. Ma, Y. Rao, Y. Cheng, C. Chen, J. Lu, and J. Zhou, "Structure-preserving super resolution with gradient guidance," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7769–7778.

[62] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "Nerf in the wild: Neural radiance fields for unconstrained photo collections," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7210–7219.

[63] Meta, "Meta Quest Pro: Our Most Advanced New VR Headset," https://www.meta.com/quest/quest-pro/, 2023.

[64] Microsoft Corporation, "High-Level Shader Language," "https://learn.microsoft.com/en-us/windows/win32/direct3dhlsl/dx-graphics-hlsl", 2023.

[65] ——, "HoloLens 2," https://www.microsoft.com/en-us/hololens/hardware, 2023.

[66] ——, "Xbox Cloud Gaming (Beta)," https://www.xbox.com/en-US/cloud-gaming, 2023.

[67] O. Mossad, K. Diab, I. Amer, and M. Hefeeda, "Deepgame: Efficient video encoding for cloud gaming," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, p. 1387–1395.

[68] Notebookcheck, "NVIDIA GeForce RTX 4090 Game Benchmarks," https://www.notebookcheck.net/NVIDIA-GeForce-RTX-4090-GPU-Benchmarks-and-Specs.674574.0.html, 2022.

[69] NP Digital, LLC, "What is Games as a Service (GaaS) and What Does it Mean For Marketers?" https://tinyurl.com/2mj9e2ms, 2023.

[70] NVIDIA Corporation, "Deep Learning Super Sampling (DLSS) Technology - NVIDIA," https://www.nvidia.com/en-us/geforce/technologies/dlss/, 2023.

[71] ——, "GeForce NOW | The Next Generation in Cloud Gaming - NVIDIA," https://www.nvidia.com/en-us/geforce-now/, 2023.

[72] Parsec Cloud, Inc., "720p Is The Preferred Broadcasting Resolution On Twitch," https://parsec.app/blog/720p-is-the-preferred-broadcasting-resolution-on-twitch-e5385a3ef08f, 2023.

[73] Playground Games Limited, "Forza Horizon 5," "https://store.steampowered.com/app/1551360/Forza_Horizon_5/", 2023.

[74] Rockstar Games, "Red Dead Redemption 2," "https://www.rockstargames.com/reddeadredemption2/", 2020.

[75] A. Romero, "Keyframes, InterFrame & Video Compression," https://blog.video.ibm.com/streaming-video-tips/keyframes-interframe-video-compression/, 2021.

[76] C. P. S.A., "The Witcher 3: Wild Hunt," "https://www.thewitcher.com/us/en/witcher3", 2023.

[77] A. Sachan, "Real-time 3D Art Best Practices - Texturing," https://cv-tricks.com/keras/understand-implement-resnets/, 2023.

[78] SAMSUNG, "Galaxy Tab S8," https://tinyurl.com/yts7x7kk, 2023.

[79] Santa Monica Studio, "God of War," "https://store.steampowered.com/app/1593500/God_of_War/", 2023.

[80] W. Sentosa, B. Chandrasekaran, P. B. Godfrey, H. Hassanieh, and B. Maggs, "{DChannel}: Accelerating Mobile Applications With Parallel High-bandwidth and Low-latency Channels," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 419–436.

[81] R. Shea, J. Liu, E. C.-H. Ngai, and Y. Cui, "Cloud gaming: architecture and performance," *IEEE network*, vol. 27, no. 4, pp. 16–21, 2013.

[82] D. Silver, "Metro Exodus," "https://store.steampowered.com/app/412020/Metro_Exodus/", 2020.

[83] Sony Interactive Entertainment LLC, "PlayStation Now," https://www.playstation.com/en-us/ps-now/, 2023.

[84] Square Enix, "Shadow of the Tomb Raider," "https://store.epicgames.com/en-US/p/shadow-of-the-tomb-raider", 2023.

[85] S. Stewart, "What Is The Best FPS For Gaming," https://www.gamingscan.com/best-fps-gaming/, 2022.

[86] Tirias Research, "Cloud Is Mobile Gaming Ready With Emergence Of Arm Infrastructure And High-Performance Video Streaming," https://www.forbes.com/sites/tiriasresearch/2020/04/14/cloud-is-mobile-gaming-ready-with-emergence-of-arm-infrastructure-and-high-performance-video-streaming/?sh=1952aff42ad6, 2020.

[87] Tobii, "Tobii Eye Tracker 5," "https://gaming.tobii.com/product/eye-tracker-5/", 2023.

[88] Topaz Labs, "Topaz Video AI," https://www.topazlabs.com/topaz-video-ai, 2023.

[89] N. Ujjainkar, J. Leng, and Y. Zhu, "ImaGen: A General Framework for Generating Memory-and Power-Efficient Image Processing Accelerators," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–13.

[90] N. Ujjainkar, E. Shahan, K. Chen, B. Duinkharjav, Q. Sun, and Y. Zhu, "Exploiting Human Color Discrimination for Memory-and Energy-Efficient Image Encoding in Virtual Reality," *arXiv preprint arXiv:2310.00441*, 2023.

[91] F. Vaccaro, M. Bertini, T. Uricchio, and A. Del Bimbo, "Fast video visual quality and resolution improvement using sr-unet," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 1221–1229.

[92] Valentin Dragoi, "Chapter 14: Visual Processing: Eye and Retina," https://nba.uth.tmc.edu/neuroscience/m/s2/chapter14.html, 2020.

[93] K. R. Vijayanagar, "How to Stop Video Buffering – 10 Effective Techniques To Reduce Buffering," https://ottverse.com/how-to-stop-video-buffering/, 2023.

[94] Vincent Tabora, "Bicubic Interpolation Techniques For Digital Imaging," https://tinyurl.com/pxm9umu9, 2020.

[95] X. Wang, L. Xie, C. Dong, and Y. Shan, "Real-esrgan: Training real-world blind super-resolution with pure synthetic data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1905–1914.

[96] J. Wei, "VGG Neural Networks: The Next Step After AlexNet," https://tinyurl.com/mpasjnzr, 2019.

[97] C.-Y. Weng, B. Curless, P. P. Srinivasan, J. T. Barron, and I. Kemelmacher-Shlizerman, "Humannerf: Free-viewpoint rendering of moving people from monocular video," in *Proceedings of the IEEE/CVF conference on computer vision and pattern Recognition*, 2022, pp. 16 210–16 220.

[98] WePC, "Video Game Industry Statistics, Trends and Data In 2023," https://www.wepc.com/news/video-game-statistics/, 2023.

[99] T. Wijman, "The Games Market in 2022: The Year in Numbers," https://newzoo.com/resources/blog/the-games-market-in-2022-the-year-in-numbers?utm_campaign=GGMR\%202022&utm_source=linkedin&utm_medium=social&utm_content=Look\%20back\%202022\%20Insights\%20Post, 2022.

[100] H. Yeo, C. J. Chong, Y. Jung, J. Ye, and D. Han, "Nemo: Enabling neural-enhanced video streaming on commodity mobile devices," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020.

[101] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han, "Neural adaptive content-aware internet video delivery," in *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, 2018, p. 645–661.

[102] H. Yeo, H. Lim, J. Kim, Y. Jung, J. Ye, and D. Han, "Neuroscaler: Neural video enhancement at scale," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, p. 795–811.

[103] Z. Ying, S. Bhuyan, Y. Kang, Y. Zhang, M. T. Kandemir, and C. R. Das, "EdgePC: Efficient deep learning analytics for point clouds on edge devices," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–14.

[104] Z. Ying, S. Zhao, S. Bhuyan, C. S. Mishra, M. T. Kandemir, and C. R. Das, "Pushing Point Cloud Compression to the Edge," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2022, pp. 282–299.

[105] Z. Ying, S. Zhao, H. Zhang, C. S. Mishra, S. Bhuyan, M. T. Kandemir, A. Sivasubramaniam, and C. R. Das, "Exploiting Frame Similarity for Efficient Inference on Edge Devices," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2022, pp. 1073–1084.

[106] M. Yoshimura, M. Kitazawa, Y. Maeda, M. Mimura, K. Tsubota, and T. Kishimoto, "Smartphone viewing distance and sleep: an experimental study utilizing motion capture technology," *Nature and science of sleep*, pp. 59–65, 2017.

[107] T. You and Y. Tang, "Visual saliency detection based on adaptive fusion of color and texture features," in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*. IEEE, 2017, pp. 2034–2039.

[108] Z. Zhan, Y. Gong, P. Zhao, G. Yuan, W. Niu, Y. Wu, T. Zhang, M. Jayaweera, D. Kaeli, B. Ren *et al.*, "Achieving on-mobile real-time super-resolution with neural architecture and pruning search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 4821–4831.

[109] H. Zhang, P. V. Rengasamy, S. Zhao, N. C. Nachiappan, A. Sivasubramaniam, M. T. Kandemir, R. Iyer, and C. R. Das, "Race-to-sleep + content caching + display caching: A recipe for energy-efficient video streaming on handhelds," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. New York, NY, USA: Association for Computing Machinery, 2017, p. 517–531.

[110] H. Zhang, S. Zhao, A. Pattnaik, M. T. Kandemir, A. Sivasubramaniam, and C. R. Das, "Distilling the essence of raw video to reduce memory usage and energy at edge devices," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. New York, NY, USA: Association for Computing Machinery, 2019, p. 657–669.

[111] L. Zhang, L. Yang, and T. Luo, "Unified saliency detection model using color and texture features," *PloS one*, 2016.

[112] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

[113] S. Zhao, P. V. Rengasamy, H. Zhang, S. Bhuyan, N. C. Nachiappan, A. Sivasubramaniam, M. T. Kandemir, and C. Das, "Understanding energy efficiency in iot app executions," in *2019 IEEE 39th international conference on distributed computing systems (ICDCS)*. IEEE, 2019, pp. 742–755.

[114] S. Zhao, H. Zhang, S. Bhuyan, C. S. Mishra, Z. Ying, M. T. Kandemir, A. Sivasubramaniam, and C. R. Das, "Déjà view: Spatio-temporal compute reuse for' energy-efficient 360° vr video streaming," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020, pp. 241–253.

[115] S. Zhao, H. Zhang, C. S. Mishra, S. Bhuyan, Z. Ying, M. T. Kandemir, A. Sivasubramaniam, and C. Das, "Holoar: On-the-fly optimization of 3d holographic processing for augmented reality," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 494–506.