Streaming of rendered content with adaptive frame rate and resolution

YARU LIU*, University of Cambridge, UK
JOSEPH G. MARCH*, University of Cambridge, UK
RAFAŁ K. MANTIUK, University of Cambridge, UK

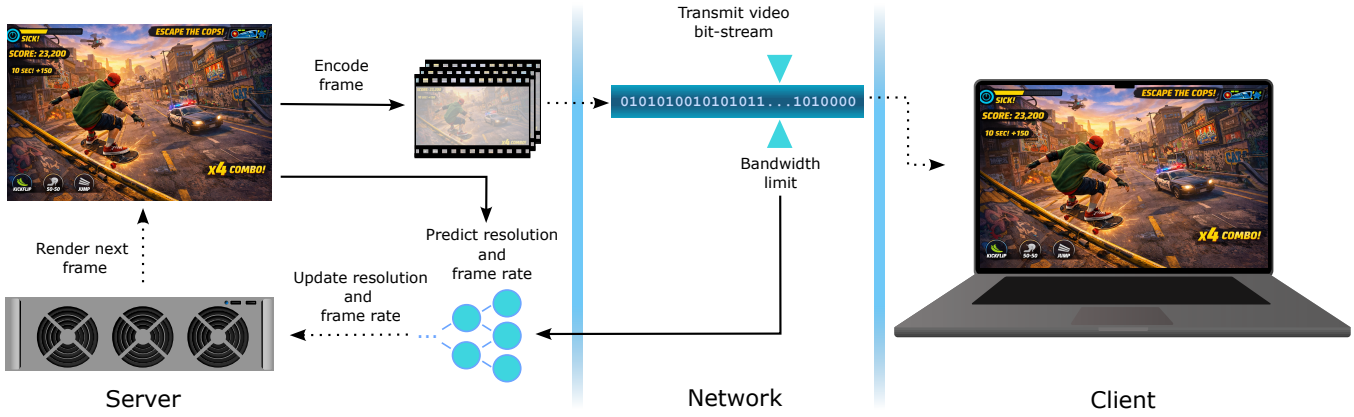


Fig. 1. Motivated by the goal of minimizing GPU usage while maintaining high visual quality, we propose a novel real-time method that leverages the human visual system to adaptively adjust the resolution and frame rate of streamed rendered content under bandwidth constraints. By jointly considering image content, motion velocity, and network bandwidth, our method predicts the resolution and frame rate combination that delivers superior perceived quality while reducing rendering costs.

Streaming rendered content is an attractive way to bring high-quality graphics to billions of mobile devices that do not have sufficient rendering power. Existing solutions render content on a server at a fixed frame rate, typically 30 or 60 frames per second, and reduce resolution when bandwidth is restricted. However, this strategy leads to suboptimal rendering quality under the bandwidth constraints. In this work, we exploit the spatio-temporal limits of the human visual system to improve perceived quality while reducing rendering costs by adaptively adjusting both frame rate and resolution based on scene content and motion. Our approach is codec-agnostic and requires only minimal modifications to existing rendering infrastructure. We propose a system in which a lightweight neural network predicts the optimal combination of frame rate and resolution for a given transmission bandwidth, content, and motion velocity. This prediction significantly enhances perceptual quality while minimizing computational cost under bandwidth constraints. The network is trained on a large dataset of rendered content labeled with a perceptual video quality metric. The dataset and further information can be found at the project web page.

CCS Concepts: • **Computing methodologies** → *Image compression*; **Perception**.

*Equal contribution

Authors' Contact Information: Yaru Liu, yl962@cam.ac.uk, University of Cambridge, Cambridge, UK; Joseph G. March, joemarch010@gmail.com, University of Cambridge, Cambridge, UK; Rafał K. Mantiuk, rafal.mantiuk@cl.cam.ac.uk, University of Cambridge, Cambridge, UK.

SIGGRAPH Conference Papers '26, Los Angeles, CA, USA

Additional Key Words and Phrases: video streaming, video quality, spatio-temporal quality, perceptual rendering, adaptive resolution rendering, adaptive frame rate rendering

ACM Reference Format:
Yaru Liu, Joseph G. March, and Rafał K. Mantiuk. 2026. Streaming of rendered content with adaptive frame rate and resolution. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '26), July 19–23, 2026, Los Angeles, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3799902.3811136

1 Introduction

Mobile devices, such as smartphones, tablets and all-in-one AR/VR headsets, have limited graphical compute power due to their energy, thermal and form-factor constraints. An attractive way to circumvent these limitations is to render frames or other relevant data on a server or PC and then stream them to the mobile device. Existing game streaming solutions typically rely on available video codecs (h.264/h.265/AV1) and mitigate network bandwidth constraints by selecting the most suitable resolution, while streaming at a fixed frame rate (commonly 60 frames per second). However, this approach results in suboptimal rendering quality under varying bandwidth and motion conditions.

A more effective solution is to adaptively select both the most suitable resolution and frame rate. This is because the fast motion found in real-time rendering can greatly benefit from higher frame rates, even if resolution must be reduced to meet bandwidth limits. Conversely, in low-motion scenarios, it is more efficient to prioritize resolution over frame rate. Fixed-frame-rate solutions cannot make this trade-off and, therefore, offer only suboptimal quality and user experience. Furthermore, by leveraging the spatio-temporal

limits of human vision, we demonstrate that rendering costs can be substantially reduced without impairing perceived quality, making our approach not only perceptually superior but also more computationally efficient.

Adaptive Frameless Rendering (AFR) [Dayal et al. 2005], reallocates samples across space and time using temporal reuse and space–time reconstruction to improve quality while minimizing computational effort. However, the approach cannot be easily adapted to modern real-time rendering pipelines. Denes et al. [2020] proposed an adaptive rendering approach in which the frame rate and resolution were selected based on motion velocity and eye-tracking data. Jindal et al. [2021] made use of variable rate shading (VRS) to apply motion-adaptive control of the local shading resolution and frame rate. However, both of these works assume that the GPU power is the main bottleneck and do not consider video streaming, instead making the naive assumption that the rendered content is constrained only by the number of pixels rendered per second. In game streaming systems, the choice of the best frame rate and resolution depends on the bandwidth of compressed video stream (in bits per second), employed video codec and multiple other variables, which have not been considered before.

Other works have proposed split-rendering approaches in which one portion of the rendering is performed on the server and another on the client [Mueller et al. 2018; Vining et al. 2025]. Split rendering can reduce the required bandwidth and latency. It requires, however, redesigned rendering pipelines, which are often incompatible with the existing game engines and require major infrastructure changes to deploy at scale.

We present a motion-aware, codec-agnostic extension to existing game streaming systems, which requires no major changes in the rendering pipeline. Our approach uses a neural network that exploits motion velocity, alongside frame content and available bandwidth, to predict the optimal encoding frame rate and resolution. A central contribution of this work is an adaptive frame rate and resolution streaming system that leverages the spatio-temporal limits of human vision and a validated video-quality metric to balance GPU rendering cost and perceptual quality under realistic network constraints.

The method yields over 50% savings in rendering cost, as measured in pixels rendered per second, while maintaining high perceived quality. The neural network runs in real time (< 2 ms) and is trained on a large dataset of rendered video clips. To label the clips, we extended the existing video quality metric, ColorVideoVDP [Mantiuk et al. 2024], to handle test and reference videos that differ in frame rate. The dataset will be released prior to publication.

Our main contributions can be summarized as:

- A method that leverages the spatio-temporal limits of human vision to adaptively select frame rate and resolution, delivering high perceptual quality in streaming while substantially reducing rendering costs.
- A large dataset[1] of 69,611 rendered with a game engine at different camera velocities, resolutions (from 360p to 1080p), frame rates (from 30 Hz to 120 Hz), and encoded at different video compression bandwidths. The dataset lets us determine the optimal frame rate and resolution at a given bandwidth and train a neural network.

[1]The dataset: https://doi.org/10.17863/CAM.129935

2 Related Work

2.1 Content-adaptive video streaming

Adaptive selection of video encoding parameters is a well-explored area of video streaming. Those methods aim at maximizing visual quality when streaming videos of varying spatio-temporal complexity under constrained or varying network conditions [Katsavounidis 2018]. Bhat et al. [2020] proposed a real-time video encoding resolution predictor. This approach relies on features extracted from the current and previous frame in combination with the resolution selected previously to compute the optimal resolution at which the current video sequence should be encoded. Spatio-temporal quality trade-off was considered in the works on the rate-distortion optimization [Vetro et al. 2001] and dynamic frame-rate selection [Thammineni et al. 2008].

Those works, however, focus on live video streaming, in which frame-rate reduction is achieved by skipping frames. In our work, we assume we can control the frame rate of the renderer, and thus we are able to stream at variable frame rates. Those works also relied on simple empirical formulas that maximized peak-signal-to-noise ratio (PSNR), which is a simple non-perceptual metric. Instead, we employ a perceptual metric (ColorVideoVDP) that models spatio-temporal vision.

2.2 No-reference metric of streamed gaming videos

The rise in popularity of game streaming services, both passive (e.g., Twitch) and active (e.g., GeForce Now), motivated research on the quality of streamed computer graphics content. This field is dominated by non-reference metrics [Barman et al. 2019; Yu et al. 2024; Zadtootaghaj et al. 2018], as reference content is difficult to acquire in quantities sufficient for training. Those metrics typically extract a number of features borrowed from existing metrics and train a machine learning model to regress those into subjective quality scores.

Yu et al. [2024] collected the LIVE-YouTube Gaming video quality dataset, comprising over 600 user-generated gaming videos and 18 600 quality ratings. They also evaluated various VQA models on this gaming database. The results indicate that natural and synthetic videos exhibit different statistical distributions, suggesting that non-reference metrics trained on natural images, such as NIQUE, may not perform well on gaming datasets. The superior performance of TLVQM compared to BRISQUE highlights the importance of incorporating motion characteristics when assessing the quality of gaming videos. Furthermore, Yu et al. highlights that deep-learning-based models can effectively capture the characteristics of synthetic videos, indicating their potential suitability for such applications.

None of the proposed metrics or datasets is suitable for our problem, as they do not model or account for the video frame rate. To address this gap, we generate a large collection of videos with reference at a range of frame rates and resolutions and extend an existing full-reference metric that models spatio-temporal vision to label our dataset.

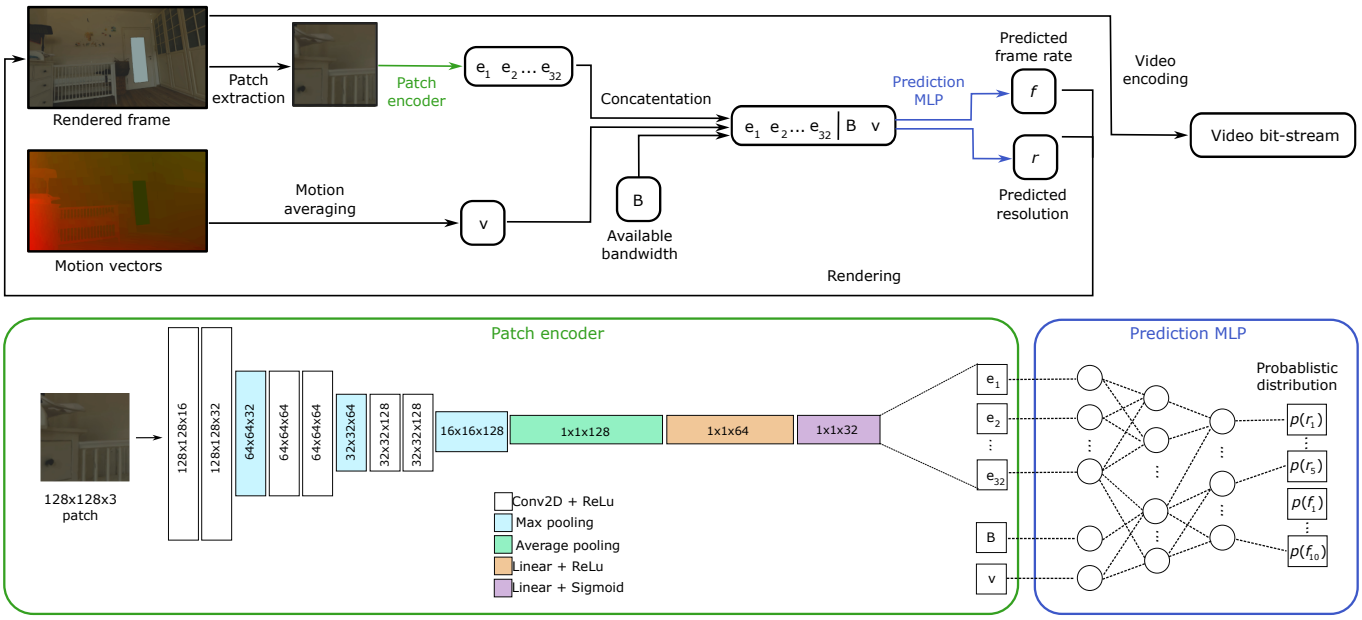


Fig. 2. Process diagram for our method. We use standard output from the rendering pipeline: frame data and motion vectors to predict the optimal resolution and frame rate for rendering and video encoding. The bottom of the figure gives architectural overviews of the patch encoder (green) and the MLP used for prediction (blue).

2.3 Streaming rendered content

Previous works designed custom rendering methods for video streaming to attain higher visual quality when streaming rendered content.

Streaming texture-space shading data has been explored as an alternative to streaming finished frames [Hladky et al. 2021; Mueller et al. 2018; Neff et al. 2022; Vining et al. 2025]. This approach offloads computationally expensive shading to the server while performing visibility calculations on the client. Recent work [Vining et al. 2025] adapted this approach to work within a fixed transmission bandwidth budget while still maintaining high image quality. While effective, such approaches require extensive modifications to typical forward-rendering pipelines. They also require all geometry data to be transferred to and then processed on the client.

Other works have demonstrated a hybrid approach of streaming limited geometry and image data [Hladky et al. 2022; Lu and Rowe 2025]. As with texture-space shading streaming, these approaches render the final image on the client device in a computationally inexpensive manner.

Liu et al. [2015] proposed a method, in which the content's depth buffer were used to compute a saliency map. The saliency map was then used to adjust the quantization parameter of each macroblock in order to redistribute the bandwidth and improve visual quality. We do not consider saliency (which can be unreliable for rendered content) and instead exploit the limitation of spatio-temporal vision to improve visual quality.

3 Streaming with adaptive frame rate and resolution

Below, we explain our approach to streaming rendered content with adaptive frame rate and resolution. Because of our focus on real-time rendering, we cannot rely on the techniques used in video streaming, such as a pre-computed resolution/bit-rate ladder [Katsavounidis 2018]. Instead, we train a real-time predictor that, given video streaming bandwidth, motion velocity, and rendered frames, can predict the optimal combination of frame rate and resolution, maximizing quality while keeping the computational cost and bandwidth limited.

The overview of our adaptive streamed rendering system is shown in Figure 2. We modified Nvidia's Falcor game engine [Kallweit et al. 2022] to encode, decode, and display the rendered frames. We randomly select a single 128×128 patch from the rendered frame and compute a moving average of frame velocity (last 500 ms) using motion vectors obtained from the G-buffer. Then, we input the patch, velocity, and current bandwidth into the neural network to predict the resolution–frame rate combination that maintains high perceptual quality while minimizing rendering cost subject to the transmission bandwidth limitation.

3.1 Real-time rendering system

To generate the dataset and test our method, we built a proof-of-concept rendering system that integrates the Falcor game engine [Kallweit et al. 2022] with real-time HEVC video encoding and decoding using NVIDIA Video Codec SDK[NVIDIA 2025a]. We render scene content to an 8-bit per channel texture and encode the result into a video stream. We then decode the frame and copy it frame-buffer, up-scaling where necessary using a bilinear filter. Both

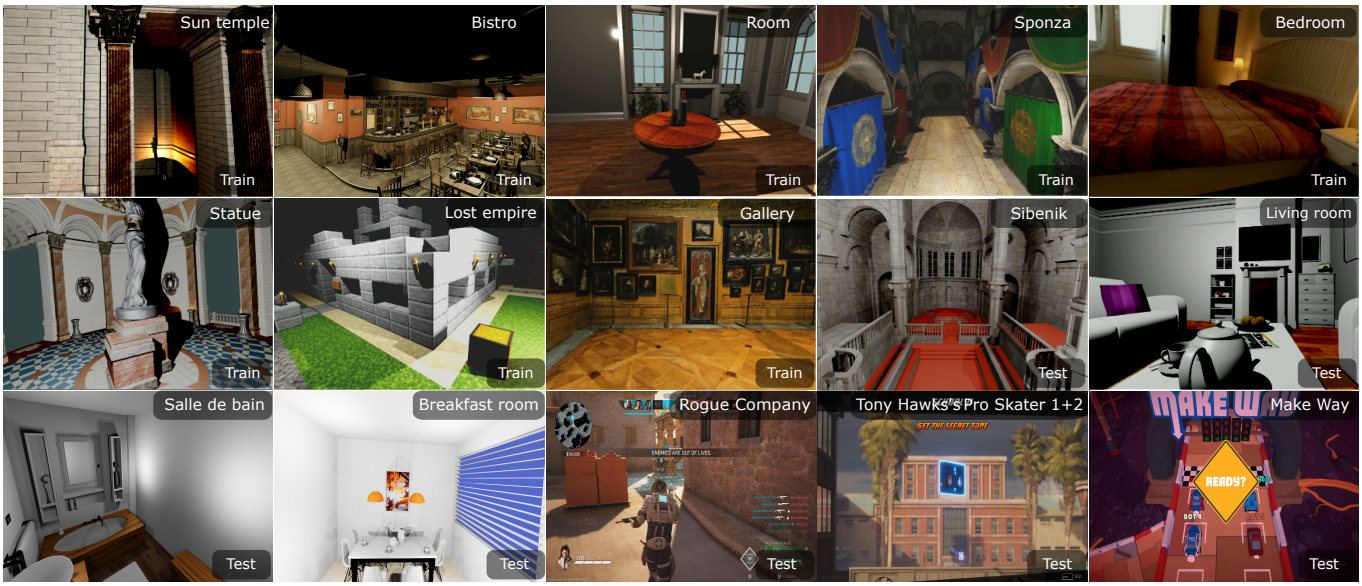


Fig. 3. Example frames from the reference videos used for training, testing, and experiments are shown. Sun temple and Statue [Games 2017] scenes were obtained from Nvidia's Open Research Content Archive [NVIDIA 2025b]. Bistro (© 2017 Amazon Lumberyard), Room (© Wig42), Sponza (© 2010 Frank Meinl, Crytek), Bedroom (© 2017 fhernand), Lost empire (© 2011 Morgan McGuire), Gallery (© 2017 The Hallwyl Museum), Living room (© 2012 Jay), Salle de bain (© Nacimus Ait Cherif) and Breakfast room (© Wig42) were downloaded from the McGuire Computer Graphics Archive [McGuire 2017], and contain only camera motion. Rogue Company (from Rogue Company by First Watch Games), Make Way (from Make Way by Ice BEAM), and Tony Hawks's Pro Skater 1+2 (by Vicarious Visions, Iron Galaxy Studios) were captured in-house from gameplay and include both camera motion and dynamic objects. Six scenes were used for testing and experiments; the simplest scene, Living Room, was excluded to keep the experiment duration manageable.

resolution and frame-rate changes are handled by reconfiguring the state of the encoder and decoder to reflect the desired parameters. Changes to resolution also force the insertion of an I-frame (intra-coded frame, without dependency on neighboring frames). To avoid stuttering during resolution changes, all necessary memory is pre-allocated and shaders are pre-compiled for each potential resolution during renderer initialization. The video decoder output resides in GPU memory, and final color conversion is handled using CUDA to obtain an image in the sRGB color space.

As our predictor only considers network bandwidth, which can be simulated by restricting encoder bandwidth, we do not implement network transmission into our proof-of-concept system.

3.2 Adaptive frame rate and resolution dataset

Training our predictor requires a large dataset. We create a dataset of representative video clips that cover a wide variety of rendered content, motion velocities, resolutions, and refresh rates. Then, we need to find the combinations of refresh rate and resolution that deliver the best quality for each clip and use them as labels for our predictor.

We use 15 scenes, shown in Figure 3, to render 3-second-long video clips. The clips are created by moving a camera along one of 15 fixed paths per scene. Twelve scenes were rendered using Falcor with static objects and moving cameras, while the remaining three were screen-recorded and featured dynamic objects. In total, thirteen scenes—ten rendered and three recorded—were used to train the predictor (see Section 3.3), while the final two rendered scenes were reserved exclusively for user studies. We used OBS (version 32.0 [Lain 2026] to record the screen when capturing footage from commercial games. The videos were encoded using the HEVC (H.265) codec, as implemented in libx265, at a constant rate factor (CRF) of 5.

For the rendered scenes used for predictor training and testing, each path is rendered at 3 motion velocities. This gives us

$10 \times 15 \times 3 = 450$ unique camera clips. Each camera path is rendered at every resolution ($r \in \{360, 480, 720, 864, 1080\}$ lines/height, 16:9 aspect ratio) and frame rate ($f \in \{30, 40, \ldots, 120\}$ Hz) combination and then encoded under one of three bit rates ($b \in \{2, 3, 4\}$ Mbps) with Nvidia's NVENC real-time codec. The three recorded scenes have 11 unique camera clips, with slow, medium, and fast velocities. The clips are encoded into 5 resolutions, 10 frame rates, and 3 bit rates as above. While it is true that encoder bitrate settings may not always be followed exactly by the compression algorithm, in our dataset, the deviation between the target and actual bitrate is negligible. Specifically, the expected error is only 0.63%, indicating that the bandwidth restrictions were effectively respected. Encoding is performed with the HEVC (H.265) codec using constant bit rate (CBR) rate control mode and the Main profile. We also generate a reference video for each camera path which is rendered at the highest resolution (1080p), 166 Hz and encoded using an NVENC encoder at a constant CRF value of 5. This gives us a total of 69,611 video clips.

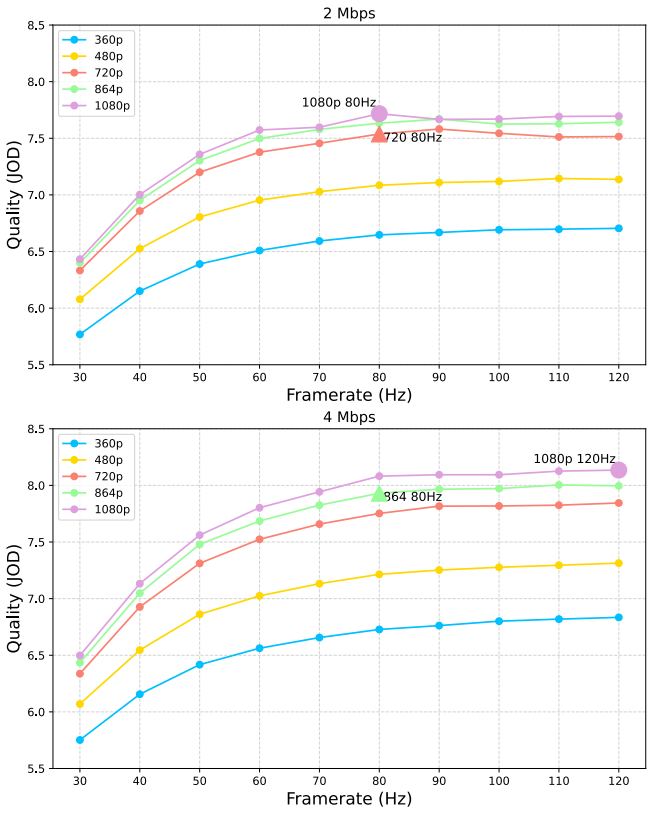


Fig. 4. ColorVideoVDP predictions for the same sequence from the scene "Bistro" rendered at different resolutions and frame rates. The two plots show the results for different bit rates. The resolution and frame giving the highest quality are shown as round markers, and the ones that also reduce the compute cost (computed from Eq. (1)) are shown as triangle markers.

We selected 1080p as the reference resolution because it remains the standard for streaming across most platforms. While 4K is supported, it is typically reserved for top-tier services and less frequently used due to bandwidth constraints. Moreover, generating and processing the 1080p dataset was computationally expensive, taking approximately a week on a GPU cluster, which made 1080p a practical upper bound for our experiments. We use 166 Hz so that none of the test frame rates can be produced by subsampling (in time) by an integer factor. Otherwise, this portion of the frames in the test sequences would coincide in time and be (almost) identical to those in the reference, giving those frame rates an unfair advantage.

Motion velocity was derived directly from the G-buffer in normalized device coordinates (NDC) by averaging the magnitudes of motion vectors across the frame. If access to the G-buffer is not possible, such motion vectors could alternatively be obtained with optical flow or by extracting them from MPEG motion vectors. It is a common practice to use videos encoded at very high bitrates (or low CRFs) as reference sequences [Chen et al. 2025; Hammou et al. 2024]. The selected CRF is, in practice, visually lossless.

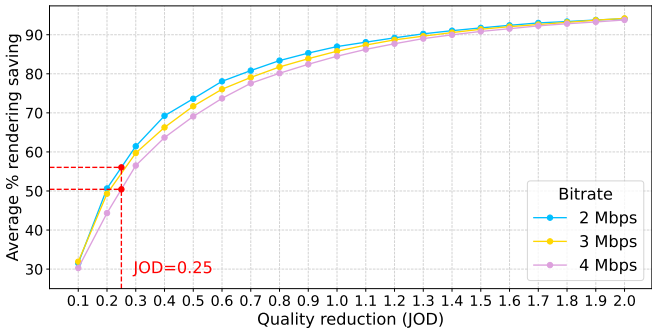


Fig. 5. Average percentage reduction in pixels rendered per second as we allow for the quality (in JODs) to drop with respect to the maximum quality (see Eq. (1)). Each curve corresponds to one of the considered bit rates.

Quality labels. The number of video clips is much too large to measure their quality in a subjective experiment. Instead, we extended a recently proposed metric modeling low-level spatio-temporal human vision, ColorVideoVDP [Mantiuk et al. 2024], to handle test and reference videos encoded at different frame rates. Our extension of ColorVideoVDP resamples the test videos to match the refresh rate of the reference video — 166 Hz. This is achieved by replicating frames between the timestamps of the original test video. Note that the replication of frames accurately simulates video that has a lower frame rate.

An example of ColorVideoVDP predictions for one scene, one path, and one velocity is shown in Figure 4 for two different bit rate settings. The plots show that, depending on the available bandwidth, the highest quality can be obtained when streaming 1080 lines at 80 Hz or 1080 lines at 120 Hz (large filled circles in Figure 4). However, we can also observe that the quality plots tend to flatten near the maximum, suggesting that selecting a slightly different resolution or refresh rate should not affect quality much. Therefore, to minimize the computation load on the server, we selected the combination of the refresh rate and resolution, which resulted in the fewest rendered pixels and was within 0.25 Just-Objectionable-Difference (JOD) units of the maximum:

$$f^*, r^* = \underset{f,r}{\operatorname{argmin}}\, f\, r^2, \quad \text{s.t.} \quad Q^* - Q(f,r) \le 0.25 \tag{1}$$

where f is the frame rate, r is the resolution, $Q(f,r)$ is the corresponding JOD quality and Q^* is the maximum quality. The choice of 0.25 JOD is validated in Section 4.1. Large triangles in Figure 4 show the resolution and refresh rate selected using this criterion. Our validation experiment (Section 4.1) will demonstrate that this choice does not affect the quality of streamed content. Figure 5 shows that allowing the quality to be reduced by just 0.25 JOD allows us to render on average 53% fewer pixels per second. We use pixels per second as a naive proxy for GPU utilization as GPU workloads are often shading bound; while this does not capture GPU compute workloads (which can be significant), it is sufficient for our purposes.

Figure 6 shows the distribution of selected refresh rates and resolutions across all clips, plotted separately for three bit rates. We can observe an expected bias towards higher frame rates and resolutions,

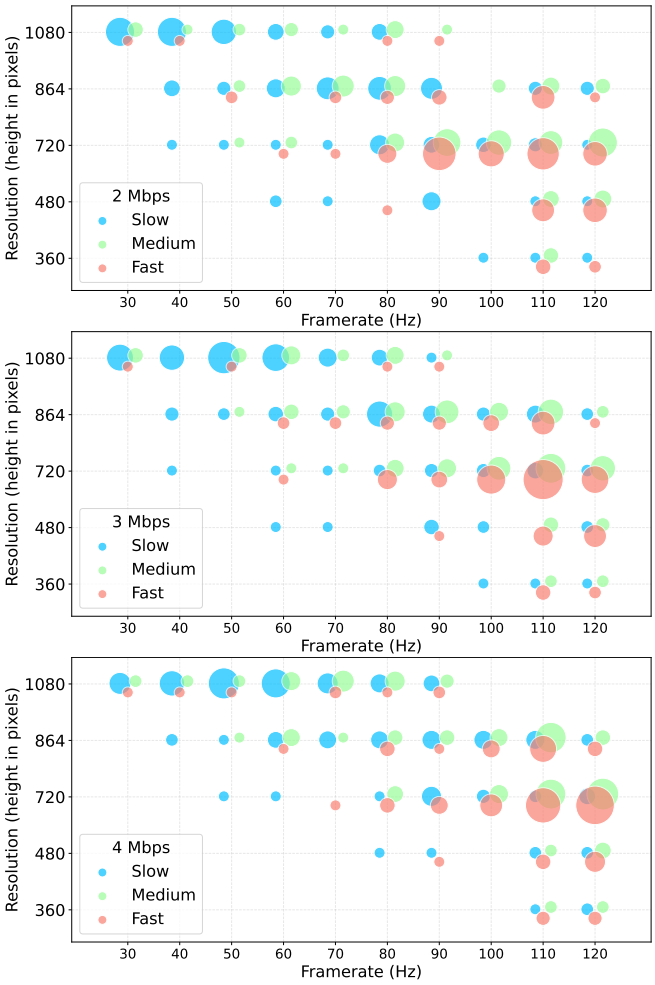


Fig. 6. Distribution of (frame rate, resolution) pairs that balance quality and efficiency, achieving quality within 0.25 JOD units of the optimal (see Eq. (1)) under 2, 3, and 4 Mbps. The position of each point is offset to better visualize density. Slower motion (blue markers) dominates at lower frame rates. The overall results suggest that a higher frame rate is not always the optimal choice.

with a higher bias at higher bandwidth. The lower refresh rates and resolutions are associated with lower velocities.

3.3 Frame rate and resolution predictor

Because of the complex input (rendered patches, velocity, bandwidth) and simple output (frame rate and resolution), the natural choice for a predictor is a neural network with a combination of convolutional and fully connected layers. Our frame rate and resolution predictor (FRRP) is a compact model with only 293,103 parameters, capable of real-time inference (2 ms). We do not need to query the predictor every frame (see Section 3.4); therefore, inference can run in a separate thread across multiple frames, which makes it suitable for integration into real-time rendering applications.

The architecture of the FRRP is shown in the bottom part of Figure 2. It consists of a patch encoder, responsible for transforming 128×128 patches into 32-value latent code vectors. It employs a 6-layer convolutional neural network (organized into three blocks of two layers each), followed by global average pooling and a 2-layer MLP with ReLU activation. A sigmoid activation in the last layer ensures that latent values are within the [0, 1] range. The latent code is concatenated with the velocity and available bandwidth and then passed to a classifier, which predicts 5 discrete resolutions and 10 discrete refresh rates, corresponding to those used in the dataset.

To compute the velocity, v, for the predictor, we first average motion magnitudes across the frame, then we use a moving average (last 500 ms) to obtain a representative value. The motion vectors are first extracted from the G-buffer in Normalized Device Coordinates (NDC), then converted to degrees per second (°/s) to make input to our predictor independent of the physical display. Because the motion distribution is heavily skewed toward low velocities, we apply a log transform to the motion input. This effectively expands low-magnitude motion while compressing high-velocity outliers. Moreover, we use 80°/s as the upper bound for motion, as the smooth pursuit eye motion (SPEM) of the human vision is unable to track higher velocities [Robinson 1964].

Training. Seven rendered and two recorded scenes were used for training, while three rendered and one recorded scene were reserved for testing (see Figure 3). We employed a 3-fold cross-validation strategy during the training to ensure model robustness. The training and testing sets comprised approximately 950,000 and 400,000 patches, respectively. We optimized a cross-entropy loss function using the Adam optimizer with a learning rate of 3×10^{-3}. Training was conducted for approximately 65 epochs with early stopping, requiring roughly 8 hours on two Nvidia RTX 4090 GPUs. The resulting confusion matrices are provided in Figure 7.

3.4 Dynamic frame rate and resolution selection

We cannot switch the frame rate and resolution too frequently for two reasons. First, most video codecs must restart the stream when the resolution is changed, which introduces additional overhead into the available bandwidth. Second, frequent changes in frame rate and resolution can be noticeable, distracting, and affect visual quality. For that reason, we use the Viterbi algorithm to ensure the best frame rate and resolution is selected every 2 seconds. We initialize the Viterbi algorithm with the transition graph weights shown in Figure 8. The weights are selected to prevent too frequent switching of the refresh rate and resolution. The predictor is run and Viterbi state is updated every frame, but the rendering frame rate and resolution are updated only every 2 seconds. A typical group of pictures (GOP) — the maximum sequence length between two I-frames — is 1 to 5 seconds. As every resolution update requires inserting an I-frame, it is desirable to align the resolution changes with the start of the GOP, which also introduces an I-frame. In our implementation, we set the HEVC group-of-pictures to 2 seconds to coincide with potential resolution changes.

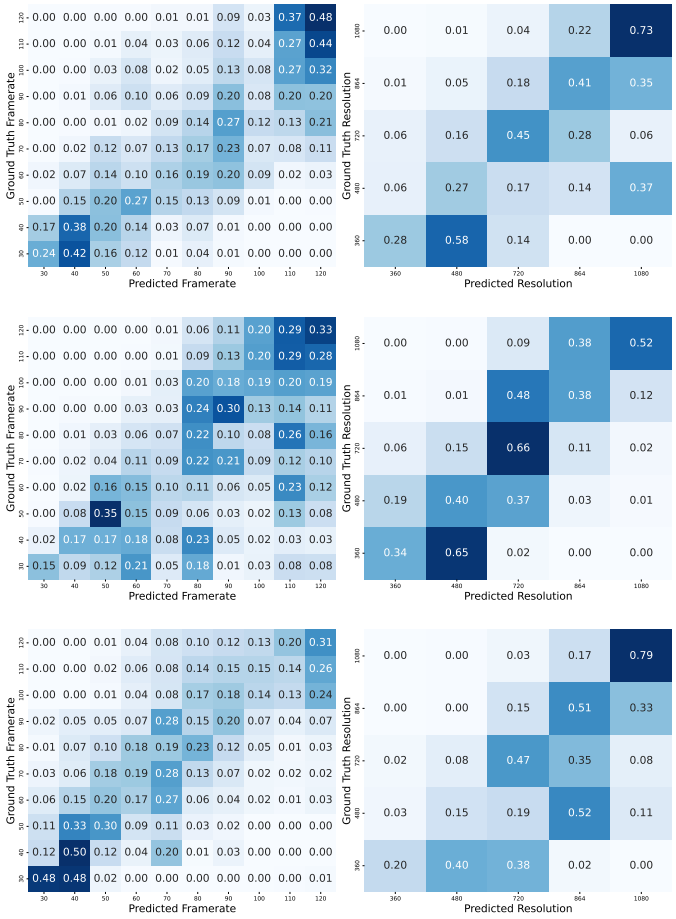


Fig. 7. Confusion matrices for frame rate and resolution prediction, showcasing our 3-fold cross-validation. These results correspond to the best-performing model configuration ("Full model" in Table 1). Each row is normalized by the total number of samples per class.

4 Validation

We validated our approach through two user studies. The first experiment demonstrates that a JOD reduction of 0.25 from the optimal configuration Eq. (1) does not show a noticeable degradation in perceived quality. The second experiment evaluates our technique against baseline approaches of streaming interactive rendered content.

Display and viewing conditions. The animations were shown on an LG OLED Evo G2 55-inch 4K display with firmware version 3.0.10. The display was connected to a workstation over a 4K HDMI 2.1 cable, which was able to transmit video content at resolutions and frame rates exceeding those used in this experiment. A Windows 11 workstation driving the display was equipped with an Nvidia GeForce RTX 4080 GPU. We conducted the experiment in a dark room. The observer viewed the content at a distance of 190 cm, corresponding to 39 pixels per degree for 1080p content.

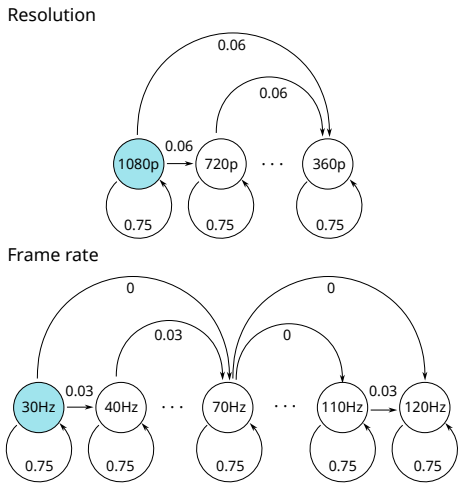


Fig. 8. Transition graphs showing the weights used for the Viterbi algorithm used to control transitions in resolution (top) and frame rate (bottom). The current state of each graph is shown in blue (1080p, 30 Hz). Note that frame rates which deviate by more than 30 Hz from the current state have a transition weight of 0, preventing any such transitions.

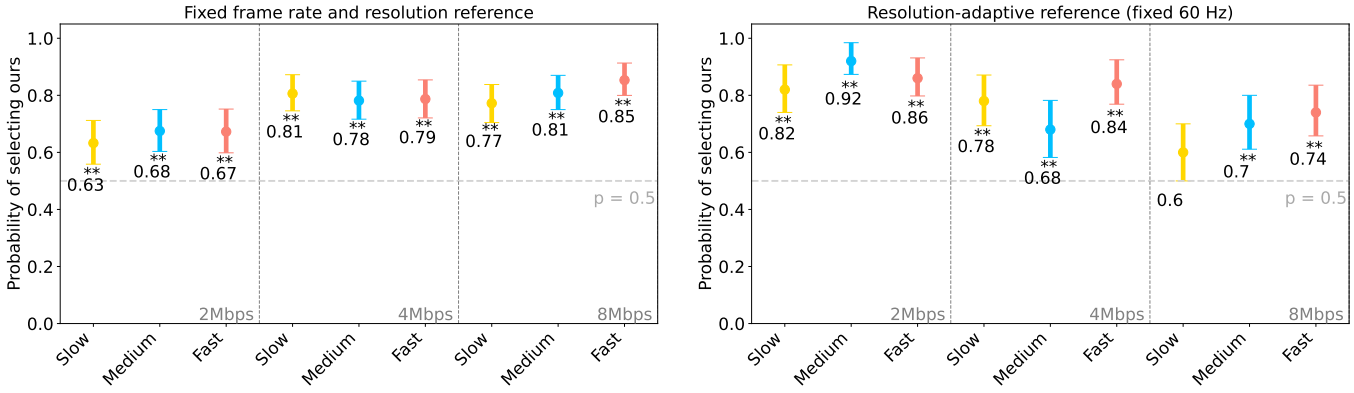
Participants. 10 observers (ages 22-32, 8 males, 2 females) participated Experiment 1. For Experiment 2, ten observers (ages 22–34; 7 males, 3 females) were recruited, including six returning participants. All observers reported that they had normal color vision and either normal or corrected-to-normal visual acuity, with some prior experience with interactive content, such as playing video games. Both studies were approved by the departmental ethics board, and participants were compensated for their time.

4.1 Experiment 1: JOD reduction

We first validate that a 0.25 JOD reduction in quality is barely perceptible, confirming this threshold does not significantly degrade perceived quality.

Stimuli. Animation sequences (3–5 s) were rendered in real-time using a modified Falcor engine following our dataset generation procedure (Section 3.2), with predefeined camera paths to ensure observer evaluated the same content. Reference videos utilized the maximum ColorVideoVDP quality configuration (round markers in Figure 4), while test videos were rendered within the 0.25 JOD margin (triangle markers). To maximize the visible differences, we selected 10 sequences exhibiting the largest $f\,r^2$ discrepancies (see Eq. (1)) between test and reference pairs.

Experimental procedure. Observers were shown sequentially pairs of test and reference videos featuring the same content and camera path in a randomized order. They were asked to pick the video with superior perceived quality — appeared sharper, had fewer distortions, and looked smoother. Both videos were shown on the same display, and observers could freely switch between them without time constraints; each switch triggered a 500 ms blank and a reset of the camera sequence and encoder state to ensure a fair comparison. Each observer completed a total of 10 comparisons.



(a) Fixed Reference

(b) Resolution-Adaptive Reference

Fig. 9. Results of the validation experiment for the two *baseline* conditions: Left — results against a fixed frame rate and resolution reference. Right — comparison against a resolution-adaptive reference. The y-axis indicates the proportion of trials in which our method was preferred over the reference videos. Error bars denote 95% confidence intervals. Results on the x-axis are grouped by motion velocity and bitrate. Asterisks mark statistical significance from a one-tailed binomial test against chance (50%): * for $p < 0.05$, ** for $p < 0.01$.

Results. Across all trials and all observers, the probability that the reference (maximum quality) is selected over the test (reduced JOD) was 0.55. Given the null hypothesis of random guess ($H_0 : \pi = 0.5$), we tested for an alternative hypothesis that the reference videos were preferred ($H_A : \pi > 0.5$). The binomial test (one-tailed, $N = 100$, $\alpha = 0.05$, $p = 0.136$) did not provide evidence for rejecting H_0, therefore, we have no evidence indicating that the videos with the JOD difference of less than 0.25 result in lower video quality. The test does not protect from Type-II errors. However, given the small effect size, achieving a statistical power of 0.8 would require more than 770 comparisons, which is impractical to collect.

4.2 Experiment 2: System validation

Second, we validate whether our technique improves the quality against constant frame rate. The experiment follows the same procedure as Experiment 1, with the following differences:

Stimuli. We tested our techniques on a diverse set of scenes, each 8–10 s long, that included static and dynamic content, varying scene complexities and motion velocities. We used all scenes from the test portion of our dataset (see Figure 3), except for the "Living room", which lacked texture complexity. Scenes "Sibenik", "Make Way", and "Rogue Company" have complex geometry, "Breakfast room", "Salle de bain", and "Pro Skater" feature flat-shaded surfaces, which are prone to banding artifacts after video encoding. "School", "Make way", and "Rogue Company" contain dynamic objects and cameras, while the others contain only camera animation. To ensure consistency, all sequences used predefined camera paths. Motion magnitude was extracted from the G-buffer for Falcor-rendered scenes and estimated via Farnebäck's dense optical flow [Farnebäck 2003] for recorded content.

We compared our technique against baselines with fixed resolution and refresh rate, and against a baseline with adaptive resolution but fixed refresh rate. The first baselines are based on the recommendations set out in [Cabrera 2025]: 1280×720 at 60 Hz for bitrates below 5 Mbps, and 1920×1080 at 60 Hz for bitrates above 5 Mbps. We chose the above configurations because these two resolutions are most commonly used for game streaming. Although 1920 × 1080, at 120 Hz can provide the best quality when streamed at high bitrates, it will also put an excessive load on the GPU, and it brings little improvement in quality at lower bitrates due to coding artifacts. Since we target"standard tier" systems, for which GPU utilization and bandwidth are a concern, 120 Hz is not a suitable baseline for us. The comparisons with those baselines included 3 scenes × 3 velocities × 3 camera paths × 3 bitrates = 81 pairwise comparisons using rendered scenes per observer. An additional 24 comparisons were collected using recorded gaming content. The second baseline was a resolution-only adaptive predictor (fixed at 60 Hz) that we trained separately. The purpose is to see if adaptively selecting both frame rate and resolution yields better results than adapting resolution alone. The comparison with this baseline included 3 scenes × 3 camera paths with different velocities × 3 bitrates = 27 comparisons for rendered content, plus 24 comparisons for recorded content.

Results. Results in Fig. 9 show that our approach improves perceptual quality across nearly all bitrates, with the clearest improvement above 2 Mbps compared to the fixed-baseline, and below 4 Mbps compared to the resolution-adaptive baseline. We speculate that this is due to the lower bit rate introducing a larger level of overall distortion into all video sequences thus making quality judgments somewhat harder to form.

5 Ablations

We evaluate the predictor's design through ablation studies, with performance results reported in Table 1. The reported errors are averaged over three train/test splits (folds), where each split included one screen-recorded and three scenes from Falcor. The remaining scenes were used for training. Those splits are separate from the train/test split used in Section 4. Table 1 shows that removing either velocity or patch information decreases performance. Velocity has

Table 1. Ablation study on input configurations. We evaluate the impact of patch size, number of patches, and velocity information on model performance. The Full model (bolded) achieves the best overall performance across all error metrics with 293,103 parameters, making it suitable for real-time applications (< 2 ms).

Model	Patch size	$n_{patches}$	Velocity	FPS error	Resolution error
w/o patch	NA	NA	✓	22%	19%
w/o velocity	128	1	✗	33%	24%
1×32×32	32	1	✓	45%	21%
1×64×64	64	1	✓	41%	18%
Full model	**128**	**1**	✓	**22%**	**17%**

a bigger impact on the frame rate prediction, while the input patch more strongly influences resolution prediction — an expected result given the spatial nature of resolution. Increasing the patch size from 32 to 128 significantly reduces prediction error, as larger spatial areas provide the network with more context to distinguish between high-frequency textures and aliasing.

The relative error metric used in Table 1 is given by the equation:

$$E = \left(\exp\left(\frac{1}{n} \sum_{i=1}^{n} \left| \log(R_{\text{test},i}) - \log(R_{\text{gt},i}) \right| \right) - 1 \right) \times 100 \qquad (2)$$

The confusion matrices for the best-performing model are shown in Figure 7. They show that in the majority of cases, the prediction error is just one class — one step in the resolution or refresh rate. Such misprediction is unlikely to cause noticeable degradation of performance.

In addition to the above ablations, we experimented with a predictor that processed multiple randomly selected patches from a frame and combined their information by average-pooling their latent vectors $\mathbf{e}$. We also tried replacing a classifier with a regressor. In both cases, the predictor's performance was comparable or worse.

6 Conclusions

Higher frame rates significantly enhance computer graphics, particularly in high-motion content. This is because they reduce latency [Spjut et al. 2019], mitigate blur and judder [Denes et al. 2020; Watson 2013], and generally improve visual quality. Our work shows that real-time game streaming services can exploit these benefits by adaptively selecting both the frame rate and the resolution.

Optimizing these parameters in real-time is challenging due to the complex interplay among factors like textures, velocity, compression artifacts, and spatio-temporal sensitivity. To address this, we successfully adapted a video metric for this task, enabling the labelling of a large dataset of 69,611 video clips, and training a neural network that predicts the optimal rendering parameters in real-time. While our primary goal was to maximize visual quality under bandwidth constraints, permitting a slight reduction in predicted quality yields substantial reductions in rendering cost. By leveraging the spatiotemporal limits of human vision, these computational savings are achieved while preserving high perceived quality.

Limitations. Modern games often mix dynamic rendered content with static user interface overlays, such as a mini-map or status information. Our method optimizes for the dynamic content and may lower the quality of user interface elements. Ideally, those should be rendered and streamed separately from the dynamic content, or our method should control shading rate rather than resolution.

Reducing GPU utilization in game streaming is often highly desirable for service providers as it can reduce overall power consumption or allow more virtual GPU (vGPU) instances to run on a single GPU. Our method also improves video quality in bandwidth-constrained scenarios. However, our solution does not address the issues caused by network latency, which can have a substantial impact on the quality of experience.

Our dataset contains mostly indoor environments without transparency, particle effects, dynamic lighting changes or stylized/non-photorealistic shading. Such content may degrade the performance of our predictor in cases where on-screen motion is not captured by motion vectors stored in the G-buffer (dynamic lighting changes, transparent particle systems, etc.).

While our modification of ColorVideoVDP proved effective at the task, as shown in two validation experiments, the metric can be further refined by retraining on a suitable dataset.

7 Acknowledgments

We would like to thank anonymous reviewers for their valuable feedback on our work. We also thank experiment participants for their contribution to this research.

References

Nabajeet Barman, Emmanuel Jammeh, Seyed Ali Ghorashi, and Maria G. Martini. 2019. No-Reference Video Quality Estimation Based on Machine Learning for Passive Gaming Video Streaming Applications. *IEEE Access* 7 (2019), 74511–74527. doi:10.1109/ACCESS.2019.2920477

Madhukar Bhat, Jean-Marc Thiesse, and Patrick Le Callet. 2020. A Case Study of Machine Learning Classifiers for Real-Time Adaptive Resolution Prediction in Video Coding. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*. 1–6. doi:10.1109/ICME46284.2020.9102934

Gerardo Delgado Cabrera. 2025. *NVIDIA NVENC OBS Guide.* https://www.nvidia.com/en-gb/geforce/guides/broadcasting-guide/

Bowen Chen, Cheng-han Lee, Yixu Chen, Zaixi Shang, Hai Wei, and Alan C. Bovik. 2025. HDRSDR-VQA: A Subjective Video Quality Dataset for HDR and SDR Comparative Evaluation. doi:10.48550/arXiv.2505.21831 arXiv:2505.21831 [cs]

Abhinav Dayal, Cliff Woolley, Benjamin Watson, and David Luebke. 2005. Adaptive Frameless Rendering. In *EGSR05: 16th Eurographics Symposium on Rendering* (2005). The Eurographics Association. https://doi.org/10.2312/EGWR/EGSR05/265-275 ISSN: 1727-3463.

Gyorgy Denes, Akshay Jindal, Aliaksei Mikhailiuk, and Rafał K. Mantiuk. 2020. A Perceptual Model of Motion Quality for Rendering with Adaptive Refresh-rate and Resolution. *ACM Transactions on Graphics* 39, 4 (Aug. 2020). doi:10.1145/3386569.3392411

Gunnar Farnebäck. 2003. Two-Frame Motion Estimation Based on Polynomial Expansion. In *Proceedings of the 13th Scandinavian conference on Image analysis* (Berlin, Heidelberg, 2003-06-29) *(SCIA'03)*. Springer-Verlag, 363–370.

Epic Games. 2017. Unreal Engine Sun Temple, Open Research Content Archive (ORCA). http://developer.nvidia.com/orca/epic-games-sun-temple

Dounia Hammou, Lukáš Krasula, Christos G. Bampis, Zhi Li, and Rafał K. Mantiuk. 2024. The Effect of Viewing Distance and Display Peak Luminance — HDR AV1 Video Streaming Quality Dataset. In *2024 16th International Conference on Quality of Multimedia Experience (QoMEX)* (2024-06). 193–199. doi:10.1109/QoMEX61742.2024.10598289 ISSN: 2472-7814.

Jozef Hladky, Hans-Peter Seidel, and Markus Steinberger. 2021. SnakeBinning: Efficient Temporally Coherent Triangle Packing for Shading Streaming. *Computer Graphics Forum* 40, 2 (2021), 475–488. doi:10.1111/cgf.14264

Jozef Hladky, Michael Stengel, Nicholas Vining, Bernhard Kerbl, Hans-Peter Seidel, and Markus Steinberger. 2022. QuadStream: A Quad-Based Scene Streaming Architecture for Novel Viewpoint Reconstruction. *ACM Transactions on Graphics* 41, 6 (Nov. 2022), 233:1–233:13. doi:10.1145/3550454.3555524

Akshay Jindal, Krzysztof Wolski, Karol Myszkowski, and Rafał K. Mantiuk. 2021. Perceptual Model for Adaptive Local Shading and Refresh Rate. *ACM Transactions on Graphics* 40, 6 (2021). doi:10.1145/3478513.3480514

Simon Kallweit, Petrik Clarberg, Craig Kolb, Tomáš Davidovič, Kai-Hwa Yao, Theresa Foley, Yong He, Lifan Wu, Lucy Chen, Tomas Akenine-Möller, Chris Wyman, Cyril Crassin, and Nir Benty. 2022. The Falcor Rendering Framework. https://github.com/NVIDIAGameWorks/Falcor https://github.com/NVIDIAGameWorks/Falcor.

Ioannis Katsavounidis. 2018. Dynamic Optimizer—A Perceptual Video Encoding Optimization Framework. *The NETFLIX Tech Blog* 4 (2018). https://netflixtechblog.com/dynamic-optimizer-a-perceptual-video-encoding-optimization-framework-e19f1e3a277f

Lain. 2026. *OBS Studio*. https://obsproject.com

Yao Liu, Sujit Dey, and Yao Lu. 2015. Enhancing Video Encoding for Cloud Gaming Using Rendering Information. *IEEE Transactions on Circuits and Systems for Video Technology* 25, 12 (2015), 1960–1974. doi:10.1109/TCSVT.2015.2450175

Edward Lu and Anthony Rowe. 2025. QUASAR: Quad-Based Adaptive Streaming And Rendering. *ACM Transactions on Graphics* 44, 4 (2025), 1–18. doi:10.1145/3731213

Rafał K. Mantiuk, Param Hanji, Maliha Ashraf, Yuta Asano, and Alexandre Chapiro. 2024. ColorVideoVDP: A visual difference predictor for image, video and display distortions. *ACM Trans. Graph.* 43, 4, Article 129 (July 2024), 20 pages. doi:10.1145/3658144

Morgan McGuire. 2017. *McGuire Computer Graphics Archive*. https://casual-effects.com/data/

Joerg H. Mueller, Philip Voglreiter, Mark Dokter, Thomas Neff, Mina Makar, Markus Steinberger, and Dieter Schmalstieg. 2018. Shading atlas streaming. *ACM Transactions on Graphics* 37, 6 (Dec. 2018), 1–16. doi:10.1145/3272127.3275087

Thomas Neff, Joerg H. Mueller, Markus Steinberger, and Dieter Schmalstieg. 2022. Meshlets and How to Shade Them: A Study on Texture-Space Shading. *Computer Graphics Forum* 41, 2 (2022), 277–287. doi:10.1111/cgf.14474

NVIDIA. 2025a. *NVIDIA Video Codec SDK*. https://developer.nvidia.com/video-codec-sdk

NVIDIA. 2025b. *ORCA*. https://developer.nvidia.com/orca

David A. Robinson. 1964. The Mechanics of Human Saccadic Eye Movement. *The Journal of Physiology* 174, 2 (Nov 1964), 245–264. doi:10.1113/jphysiol.1964.sp007485

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*. ACM, Brisbane QLD Australia, 110–113. doi:10.1145/3355088.3365170

Arunoday Thammineni, Arvind Raman, Sarat Chandra Vadapalli, and Sriram Sethuraman. 2008. Dynamic Frame-Rate Selection for Live LBR Video Encoders Using Trial Frames. In *2008 IEEE International Conference on Multimedia and Expo*. 817–820. doi:10.1109/ICME.2008.4607560

A. Vetro, Yao Wang, and Huifang Sun. 2001. Rate-Distortion Optimized Video Coding Considering Frameskip. In *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*, Vol. 2. IEEE, Thessaloniki, Greece, 534–537. doi:10.1109/ICIP.2001.958169

N. Vining, Z. Majercik, F. Gu, T. Takikawa, T. Trusty, P. Lalonde, M. McGuire, and A. Sheffer. 2025. FastAtlas: Real-Time Compact Atlases for Texture Space Shading. *Computer Graphics Forum* (April 2025), e70010. doi:10.1111/cgf.70010

A. B. Watson. 2013. High Frame Rates and Human Vision: A View Through the Window of Visibility. *SMPTE Motion Imaging Journal* 122, 2 (March 2013), 18–32. doi:10.5594/j18266

Xiangxu Yu, Zhenqiang Ying, Neil Birkbeck, Yilin Wang, Balu Adsumilli, and Alan C. Bovik. 2024. Subjective and Objective Analysis of Streamed Gaming Videos. *IEEE Transactions on Games* 16, 2 (2024), 445–458. doi:10.1109/TG.2023.3293093

Saman Zadtootaghaj, Nabajeet Barman, Steven Schmidt, Maria G. Martini, and Sebastian Möller. 2018. NR-GVQM: A No Reference Gaming Video Quality Metric. In *2018 IEEE International Symposium on Multimedia (ISM)* (2018-12). 131–134. doi:10.1109/ISM.2018.00031