MNSS: Neural Supersampling Framework for Real-Time Rendering on Mobile Devices

Sipeng Yang, Yunlu Zhao, Yuzhe Luo, He Wang, *Member, IEEE,* Hongyu Sun, *Member, IEEE,* Chen Li, Binghuang Cai, Xiaogang Jin*, *Member, IEEE*

Abstract—Although neural supersampling has achieved great success in various applications for improving image quality, it is still difficult to apply it to a wide range of real-time rendering applications due to the high computational power demand. Most existing methods are computationally expensive and require high-performance hardware, preventing their use on platforms with limited hardware, such as smartphones. To this end, we propose a new supersampling framework for real-time rendering applications to reconstruct a high-quality image out of a low-resolution one, which is sufficiently lightweight to run on smartphones within a real-time budget. Our model takes as input the renderer-generated low resolution content and produces high resolution and anti-aliased results. To maximize sampling efficiency, we propose using an alternate sub-pixel sample pattern during the rasterization process. This allows us to create a relatively small reconstruction model while maintaining high image quality. By accumulating new samples into a high-resolution history buffer, an efficient history check and re-usage scheme is introduced to improve temporal stability. To our knowledge, this is the first research in pushing real-time neural supersampling on mobile devices. Due to the absence of training data, we present a new dataset containing 57 training and test sequences from three game scenes. Furthermore, based on the rendered motion vectors and a visual perception study, we introduce a new metric called inter-frame structural similarity (IF-SSIM) to quantitatively measure the temporal stability of rendered videos. Extensive evaluations demonstrate that our supersampling model outperforms existing or alternative solutions in both performance and temporal stability.

Index Terms—Neural supersampling, deep learning, real-time rendering

1 Introduction

The required amounts of computational resources for real-time rendering have been growing significantly to meet the demands of higher resolution, higher refresh rate, and modern rendering techniques such as real-time ray tracing and various global illumination techniques. In particular, the popularization of smartphone screens with 1080P or 2K resolution and 90 Hz refresh rate poses a challenge for mobile real-time rendering which is often limited by hardware capability and battery power budget. Consumers and developers have to make compromises among the rendering quality, image resolution, and frame rate [1].

Recent research on video supersampling has shown that it is possible to reconstruct higher resolution videos with realistic details using neural networks [2], [3]. This has attracted attention in several communities, including computer graphics [4], [5], where how to render high-quality frames faster is a fundamental research topic. Neural supersampling methods for real-time rendering aim to reduce the rendering cost while preserving image quality. They first render frames at low-resolution (LR), which can significantly reduce the shading computational overhead, and then upscale the LR frames to high-resolution (HR) through efficient neural networks. Prior works [4], [6], [7] have proven the feasibility of neural supersampling for real-time rendering. These methods, however, are not suitable for a wide range of hardware platforms because high performance computing hardware is required. Deep learning supersampling (DLSS) 2.0 [7] can reconstruct a 4K resolution image in under 2ms using a lower resolution input, but it can only be run on high-end GPUs with NVIDIA RTX Tensor Cores. Neural supersampling for real-time rendering (NSRR) [4] introduces a hardware-independent approach

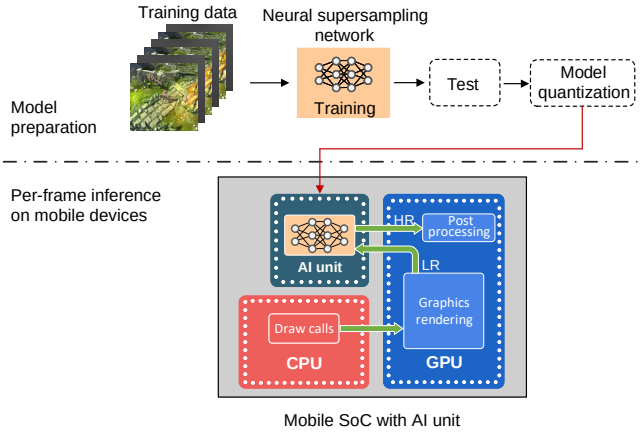


Fig. 1. A high-level overview of our approach. In the preparation phase (top), we collect paired low resolution and high resolution data to train the neural supersampling network. After testing and performance evaluation, the model parameters are quantized to 8-bit integers to reduce computational costs. For the implementation on mobile devices, the quantized model is loaded into the AI unit to increase the image resolution during rendering.

- *Sipeng Yang, Yunlu Zhao, Yuzhe Luo, and Xiaogang Jin are with State Key Lab of CADCG, Zhejiang University, Hangzhou, PR China.*
- *He Wang is with the School of Computing, University of Leeds, Leeds LS2 9JT, UK.*
- *Hongyu Sun, Chen Li, and Binghuang Cai are with OPPO US Research Center, Bellevue, WA, USA.*
- *Xiaogang Jin is the corresponding author. E-mail: jin@cad.zju.edu.cn*

Manuscript received MM XX, 2022; revised MM XX, 2022.

based on U-Net, which takes approximately 20ms to reconstruct a 1080p image on a GPU with NVIDIA TensorRT optimization. However, it requires large memory resources that incur high costs, and its speed is low because it requires extensive computations.

Compared to prior work, neural supersampling for real-time rendering applications on mobile devices is significantly more challenging. First, using many input frames like NSRR [4] is infeasible on mobile devices due to limited memory. Further, when aiming for real-time applications, the performance requirement is even stricter than PCs due to mobile devices' limiting computation budget. Naively reducing the size of existing models to boost performance will lead to severe quality deterioration.

To address the aforementioned challenges on mobile devices, we propose an efficient mobile neural supersampling framework (MNSS) capable of reconstructing high-fidelity images at a target resolution while requiring much lower computation cost than existing approaches. To reduce memory cost, we recurrently reuse only one previous frame. A sub-pixel sample pattern is applied to maximize sampling efficiency and preserve details. The final result is obtained by exploiting neural networks to appropriately blend the history frame with the new samples of the current frame. Fig. 1 illustrates the high-level overview of our method. We first collect paired LR and HR video data to train the neural supersampling network, and then we evaluate its performance and quantize it via a post-training quantization scheme [8]. For the inference phase on mobile phones, we deploy the quantized supersampling model into the applications' rendering pipelines. Many mobile system-on-chips (SoCs) have embedded dedicated artificial intelligence (AI) units in recent years, such as Qualcomm Hexagon digital signal processors (DSPs) and MediaTek AI processing units (APUs). These units are designed to perform fast matrix computation while consuming little power. To that end, we designed the neural supersampling network to run on these low-power AI units.

In rendering, motion vectors capture the per-pixel screen-space motion from one frame to the next. They can be accurately calculated in modern renderers. Based on the accurate motion information, we introduce a new metric to quantitatively measure the temporal consistency of rendered videos, called inter-frame structural similarity (IF-SSIM). It uses the ground-truth motion vectors to align the frames of the test videos, and then the temporal stability is represented as the similarity between adjacent frames. Besides, the ground-truth depth maps and a perceptual difference threshold are used to eliminate inherent disocclusions and shading changes among frames.

To our best knowledge, there is no public dataset for rendering neural supersampling. Therefore, we build a new dataset using Unity Engine [9]. Color images, depth maps, and motion vectors are collected from three dynamic game scenes. Based on this dataset, we are able to train and test supersampling methods on real application scenarios, including our approach, NSRR [4], and existing SR methods [2], [3], [10], [11]. We validate the superior performance of our method by comparing it to the SOTA baseline and SR approaches, extensive experiments demonstrate that our method achieves better image quality and temporal stability while requiring relatively less running time.

We summarize our main contributions as follows:

- An efficient neural supersampling framework optimized for real-time rendering, which can reach nearly 90 FPS on a mobile device equipped with an AI unit. To our best knowledge, this is the first research in pushing rendering neural supersampling on smartphones.
- A new metric, IF-SSIM, uses rendered motion vectors to quantitatively evaluate videos' temporal stability.
- A public dataset, *GameVideo57*, containing 57 rendered videos and auxiliary buffers (a total of 73,000 frames).

The remainder of the paper is organized as follows. We review the related work in Sec. 2. The proposed supersampling framework, the temporal stability metric IF-SSIM, and the dataset are respectively described in Sec. 3, 4, and 5. The results and qualitative evaluations are presented in Sec. 6. Finally, we conclude the paper in Sec. 7.

2 Related Work

2.1 Real-Time Rendering

Antialiasing. Antialiasing is a fundamental problem in computer graphics. In practice, real-time applications are limited to a low sampling rate, typically one sample per pixel, resulting in the undersampling problem [12]. Undersampling will cause artifacts such as jagged edges, spatial noise, flickering, etc. Antialiasing techniques [13], [14], [15] aims to remove such undersampling artifacts, which can be naturally achieved by increasing the per-pixel sampling rate using supersampling [16].

Temporal Supersampling. Temporal supersampling is based on a simple observation. That is, most of the on-screen content does not change between adjacent frames. One of the most representative temporal supersampling techniques is temporal antialiasing (TAA) [17], which is becoming increasingly important for various real-time applications. We refer readers to [18] for a comprehensive survey on related temporal algorithms [19], [20], [21]. As temporal supersampling often suffers from temporal artifacts (such as flickering and blurriness) introduced by the heuristic algorithms they used, researchers are trying to further improve the result by using learning-based approaches. One recent trend in leveraging temporal data is to reduce the rendering cost by using learned supersampling methods. Deep learning supersampling (DLSS) 2.0 [7], developed by NVIDIA, has good performance and produces decent quality results. However, their network structure and implementation details are not open to the public. Xiao *et al.* [4] proposed a hardware-independent neural supersampling method that uses a U-Net to reconstruct high resolution frames. On a high-end GPU with NVIDIA TensorRT, it takes about 20ms to generate one 1080p frame, which is relatively slow for real-time rendering. Thomas *et al.* [22] present a quantized model QW-Net for image reconstruction because quantization techniques can significantly reduce both the memory requirement and the computational overhead of using neural networks. Most QW-Net computations, in particular, are implemented with 4-bit integers and accelerated

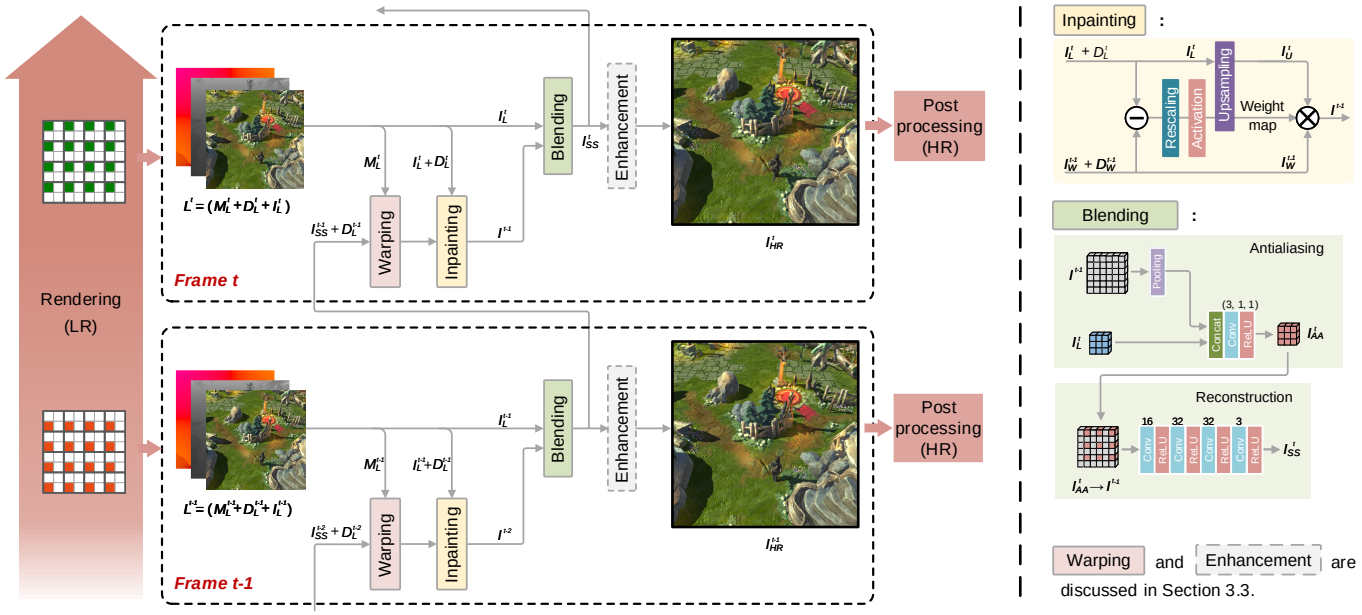


Fig. 2. Overview of our proposed neural supersampling framework. The left shows the pipeline of the method, and the right shows the architecture of sub-networks. For current *Frame t*, we first render the LR data L^t by adding a viewport sub-pixel offset to the camera. Then, the previous reconstructed frame I_{SS}^{t-1} and its depth map D_L^{t-1} are loaded and reprojected to align to the current frame using the motion information M_L^t, following which a weight map is generated by inpainting module to fill in invalid history pixels. After that, the current frame I_L^t and the repaired history frame I^{t-1} are fed into the blending network to generate HR output I_{SS}^t. In addition, the enhancement module can be optionally active by the user to sharpen edges. Lastly, the reconstructed frame is pulled through the post-processing stage of the rendering pipeline.

using dedicated hardware. This network is used in a frame-recurrent approach to improve image quality for real-time rendering, which outperforms TAA. However, their method is infeasible for the mobile platform because mobile devices have far less computing power than PCs. Concurrent to our work, AMD presents two non-machine-learning solutions, FidelityFX super resolution (FSR) 1.0 [23] and 2.0 [24], which utilize cutting-edge upscaling techniques to achieve high-quality output.

2.2 Super-Resolution

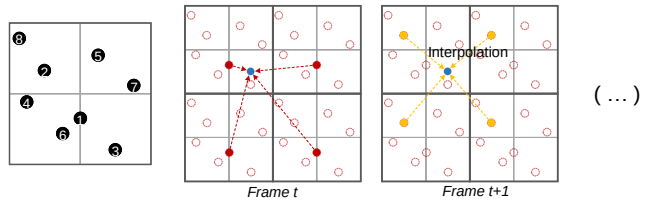
Super-resolution (SR) aims at reconstructing high-resolution (HR) output from low-resolution (LR) images or videos. Since the pioneer's work SRCNN [25], a lot of deep-learning-based single image super-resolution (SISR) methods [2], [26], [27], [28] and Video super-resolution (VSR) methods [2], [29], [30] have been proposed.

Single Image Super-Resolution. Increasing resolution while recovering more details is the key to SISR [31]. In recent years, convolutional neural networks based approaches [25], [26], [32], [33] have dominated the research of SISR and achieve high quality SR results due to their strong feature extraction and representation capabilities. Most of these methods are fully supervised. They usually obtain paired LR-HR training data by degrading HR images to LR images. Aside from fully supervised approaches, some work [25], [26], [32], [33] adopt generative adversarial networks to generate HR images. These methods can generate more realistic results in real-world cases because of avoiding manual image degradation [31].

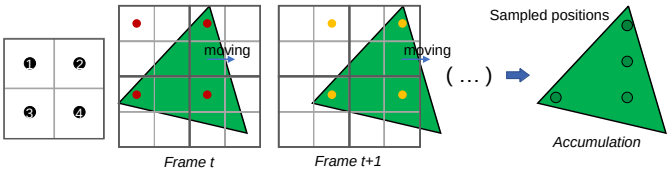
Video Super-Resolution. VSR focuses on reconstructing the HR frame from a series of adjacent LR frames. One of the main differences among learning-based VSR methods is how to use the information of history frames [34]. Some approaches [3], [35], [36], [37] use alignment-based algorithms to capture the information of adjacent frames, where optical flows [38] and deformable convolution [39] are the common practices. While some work [40], [41], [42] exploit the temporal feature extraction power of RNNs. They directly use the original LR images and share the spatio-temporal information between neighboring images to reconstruct HR frames. In particular, aiming at mobile platforms, EVSRNet [11] presents a lightweight VSR model that can be applied on mobile devices. It adopts neural architecture search techniques to find the optimal VSR model.

2.3 Temporal Stability Assessment

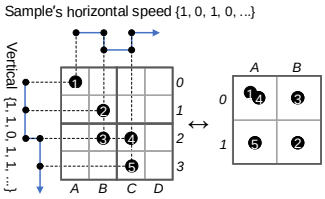
Temporal stability is one of the most important factors in video quality assessment. Ghosting, flickering, and aliasing artifacts are the common instability causes in rendered videos. There have been some studies that take into account temporal stability in video quality assessment. The spatio-temporal reduced reference entropic differences (STRRED) [43] and its variants utilize the wavelet coefficients of frames to measure the differences between the reference and the distorted videos. Learning-based approaches, such as VMAF [44], usually extract temporal and spatial features and then calculate the quality score via regression. Some works incorporate psychophysical studies of the human visual system in video quality and temporal stability assessment to evaluate the visual impact of spatial and temporal distortions. To detect visible distortions, contrast sensitivity function (CSF) [45] based models are typically used. 3D CSF [46] and FovVideoVDP [47], for example, use spatio-temporal CSF to determine perceptible



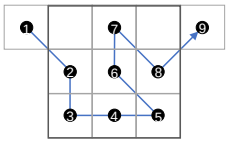
(a) Halton sampling pattern and the subpixel calculation process



(b) Naive pixel-by-pixel sampling pattern and the subpixel sampling process



(c) Our ×2 sampling pattern



(d) Our ×3 sampling pattern

Fig. 3. Conceptual illustration of Halton pattern, naive pixel-by-pixel sampling pattern, and our sampling patterns. For $\times 2$ upscaling, a 2×2 pixel tile represents one pixel of LR images. (a) shows the Halton sampling pattern and and the subpixel calculation process. Given the samples (red points) of *Frame t*, the color of each subpixel is obtained by bicubic interpolation of the surrounding samples, which would introduce resampling blur. (b) shows the sampling process of a naive pixel-by-pixel sampling pattern. When the object (the green triangle) is moving at the speed of sampling positions, the temporal samples will overlap in some places. Our alternate sub-pixel sampling patterns are shown in (c) and (d). The samples of our pattern are moving at variable speeds in the horizontal and vertical directions. It ensures the samples do not move with any object in the 3D scene.

spatial and temporal frequencies. The distortion maps and quality values can then be calculated using a psychometric function or a contrast masking method. However, the above methods do not decouple the image quality (*i.e.*, similarity to references) and the temporal stability of frame sequences, because they still measure by calculating differences between the distorted video and its reference.

3 Method

3.1 Overview

Our main goal is to generate high-fidelity HR frames from the low sampling rate rendered inputs for mobile real-time rendering applications. To this end, we propose an efficient mobile neural supersampling (MNSS) framework that can integrate into the rendering pipeline of graphics applications. It uses an alternate sub-pixel sample pattern to improve the sampling efficiency and adopts a frame-recurrent approach to recover fine image details.

Fig. 2 illustrates the pipeline of our method which corresponds to the neural supersampling network presented in Fig. 1. For *Frame t*, the input of our model is LR data L^t, including color image I_L^t, depth image D_L^t, and motion vectors M_L^t, rendered in tone-mapped space. We propose an alternate sub-pixel sample pattern tailored for integral upscaling factors (Sec. 3.2), that can significantly improve the sampling efficiency. The input color images and depth images are rendered with our proposed sub-pixel sample pattern. For image reconstruction networks, we start with a frame-recurrent approach [2], [48], that recurrently warps and accumulates history frames. First, the history data I_{SS}^{t-1} and D_L^{t-1} are loaded and reprojected to I_W^{t-1} and D_W^{t-1} to align to the current frame using the motion information. Then, a weight map generated in the inpainting module is used to check and reject invalid pixels of the history frame I_W^{t-1}, and pixels from the upsampled current frame I_U^t are used to rectify and fill in the invalid history pixels. Lastly, we design lightweight blending networks to perform antialiasing and reconstruct HR frame I_{SS}^t using the current LR frame I_L^t and the history HR frame I^{t-1}.

3.2 Sub-pixel Sampling Pattern

Despite some effective low-discrepancy sampling sequences have been proposed and used in rendering sampling tasks, such as Halton and Sobol sequences, they are not ideal for our method. As shown in Fig. 3 (a), when HR images are rendered at lower sampling rates using the Halton pattern, severe resampling blur may occur due to the interpolation operation. While naive pixel-by-pixel sampling pattern may cause samples overlapping when objects and sampling points are moving at the same speed (see Fig. 3 (b)). From a theoretical standpoint, a random sampling pattern is feasible but not the best, due to the momentary samples overlapping problem as Fig. 3 (b) may emerged in some short random sequences.

To maximize sampling efficiency, we propose a new sub-pixel sample pattern designed for integral upscaling supersampling. We shade samples at the center of each subpixel to avoid introducing resampling blur and use a variable speed jittering pattern to deal with the overlapping problem (where a subpixel refers to one pixel of each 2×2 or 3×3 pixel tile). For $\times 2$ upscaling, the sampling pattern is defined as the sampling order of 4 subpixels. Due to the pixel tiles being connected to each other, regular sampling patterns often cause biased results (blurry or aliased). Some objects in the scene can easily move in sync with the sampling points, which would cause the samples overlapping problem. Therefore, we introduce a variable speed jittering pattern to avoid these potential issues. Our method periodically adjusts the motion speeds of sampling points rather than the sampling positions and uses different cycles in horizontal and vertical to simulate complex motions. Fig. 3 (c) gives a feasible trajectory follows the above rules, where the horizontal and vertical speed cycles are $\{1, 0\}$ and $\{1, 1, 0\}$ (in the unit of pixel widths per frame). Theoretically, other speed cycles that follow the above rules are also practicable to construct the sampling pattern.

Sampling patterns for $\times 3$ upscaling supersampling are similar. To evenly scan pixel tiles in shortest time, we design a fine sampling pattern as shown in Fig. 3 (d) which cover the 3×3 pixel tiles over every 9 frames. It is worth noting that, to balance rendering time and image quality, $\times 2$ or $\times 3$ upscaling factor is sufficient for neural supersampling tasks. For $\times 4$ and above factors supersampling, the image quality

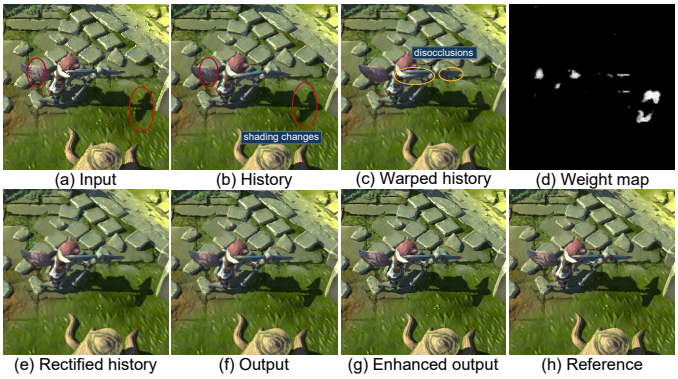


Fig. 4. Intermediate images of our method.

is severely degraded but with limited performance gains. We can adjust the performance by modifying the depth and width of the blending network instead of using a lower sampling rate.

The implementation of our method only requires few engine code changes. In practice, sampling at sub-pixel positions can be achieved by adding a jitter offset to the camera projection matrix. Taking the sampling pattern of $\times 2$ upscaling for example, we first set the rendering resolution to half of the target resolution and adjust the mip-map level bias of the texture to match the level of target resolution images. Then, we add a jitter offset to the camera projection matrix to render the first frame, and change the offset according to the designed sampling pattern when rendering the following frames. To evaluate the performance gains from the proposed sub-pixel sampling pattern, we gives the sampling efficiency comparisons in Sec. 6.4.

3.3 Reconstruction Framework

After finishing LR sub-pixel rendering, we adopt a frame-recurrent based network to reconstruct high-fidelity HR frames. As shown in Fig. 2, the reconstruction network consists of history frame warping and inpainting, multi-frame blending, and enhancement modules. We describe the detailed structure of each module in this section.

3.3.1 Warping Module

We first project the supersampling result of the previous frame to the current using backward warping with bilinear interpolation. Warping operation allows for filtering history pixels in a small receptive field to deal with a large motion vector, which is also widely used in VSR methods [34]. To obtain the motion information of objects between adjacent frames, VSR methods usually use estimated optical flows, while our method uses rendered motion vectors that can be accurately calculated in modern renderers. Each element of the motion vectors represents the image space offset between the pixel's current location and its location in the previous frame. In Fig. 4 (b) and (c), we show an example of the history and the warped history frame. The warped history frame is obtained by:

$$I_W^{t-1} = f_W(I_{SS}^{t-1}, M_U^t), \quad D_W^{t-1} = f_W(D_U^{t-1}, M_U^t), \qquad (1)$$

where $f_W(\cdot)$ represents the backward warping operation, I_{SS}^{t-1} is the supersampling result of the previous frame, and D_U^{t-1} is the upsampled depth image of the previous frame, M_U^t is the upsampled motion vectors of the current frame.

3.3.2 Inpainting Module

The warped history frame I_W^{t-1} still cannot be directly used in the blending network. Sudden changes in occlusion or lighting will make some pixels invalid for reuse. Using history samples without verification will lead to serious ghosting artifacts [18]. In previous approaches, NSRR [4] uses a weight map to adjust the values of history frame features to reduce the impact of invalid history pixels. It requires an extremely heavy neural network to generate the weight map for each previous frame. Zeng *et al.* [49] propose temporally reliable motion vectors that track the movement of the disocclusion regions to address the ghosting issue. But calculating the new motion vectors will involve a lot of extra computation which is not friendly to mobile devices.

Inspired by the history samples accumulation process in TAA algorithms [18], we propose an efficient inpainting network to deal with the stale or invalid history pixels. As shown in Fig. 2, the inputs of the inpainting module include the current and previous color images and depth maps. We first calculate a difference map between the current and the previous color and depth frames. Because the current frame is sampled at a low resolution, the difference map is calculated using only the corresponding subpixels of the previous frames I_W^{t-1} and D_W^{t-1}. Then a learned 3×3 convolution kernel with bias, a nonlinear activation function, and an upsampling layer are used to rescale and project the different map to the expected weight map M_w. For the nonlinear activation function, the sigmoid or the tanh function (clipped to [0, 1]) that can provide a smooth transition between the frames is suited for our method. After that the weight map and the current frame are upsampled to the target resolution to rectify invalid history pixels:

$$I^{t-1} = M_w \cdot I_U^t + (1 - M_w) \cdot I_W^{t-1}, \qquad (2)$$

where I^{t-1} is the rectified history frame. Each pixel of the weight map M_w represents the probability of the history pixel for reusing. Fig 4 provides an example of the rectification process. After training, the inpainting module can generate effective weight maps to the invalid history pixels with the current samples (see Fig. 4 (c), (d), and (e)).

Note that, for the upsampling process, to avoid spatial offset introduced by our sampling pattern, we place the pixels of LR images at the sampling positions of HR image I_U^t. And other pixels of the HR image I_U^t are filled in by bilinear interpolation.

3.3.3 Blending and Enhancement Module

We aim to use neural networks to blend current and previous frames to reconstruct high-quality images. Moreover, the neural networks need to be deployed on mobile devices and run in real time. Therefore, we design a lightweight blending module comprised of an antialiasing and a reconstruction network. Since our model works at an extremely low sampling rate, the sampled LR frames often suffer from spatial aliasing artifacts. To integrate antialiasing into our model, we rectify the current frame samples using the previous frame first. Given a rendered current frame I_L^t and

its rectified previous HR frame I^{t-1}, I^{t-1} first goes through an average pooling layer that resizes the image to LR. Then the pooled data and I_L^t are concatenated to generate anti-aliased output I_{AA}^t by a low complexity neural network. For the reconstruction network, we insert the pixels of the anti-aliased LR frame I_{AA}^t into the rectified previous frame I^{t-1} according to the sampling positions. This combination is fed to a wider and deeper neural network to construct high-fidelity HR current frame I_{SS}^t, as shown in Fig. 2. The numbers of input and output channels of the reconstruction network are given in the figure.

Image enhancement algorithms can effectively improve visual effects. Thus, our framework provides an optional image enhancement module, which uses a sharpening algorithm to make edges appear sufficiently pronounced to improve image quality further. We apply the unsharp masking [50] to the reconstructed image I_{SS}^t using a sharpen filter kernel $\begin{bmatrix} -\beta & 0 & -\beta \\ 0 & 1+(4\times\beta) & 0 \\ -\beta & 0 & -\beta \end{bmatrix}$, where β is an adjustable parameter. Larger β gives sharper images, and the recommended value of β is 0.15.

3.4 Loss Function and Training Tricks

We use a combined loss function to train our networks, which considers the structural and perceptual loss of HR supersampling results and the per-pixel loss of LR anti-aliased frames:

$$Loss = L_{struct} + L_{percept} + L_{AA}, \tag{3}$$

where

$$\begin{cases} L_{struct} = 1 - \text{SSIM}\left(I_{SS},\, I_{GT}\right) \\ L_{percept} = w \cdot \text{MSE}\left(\phi\left(I_{SS}\right), \phi\left(I_{GT}\right)\right) \\ L_{AA} = k \cdot \text{mean}(\|I_{AA} - I'_{GT}\|)), \end{cases}$$

I_{SS} and I_{GT} are the reconstructed image and the reference image, I_{AA} is LR anti-aliased images, I'_{GT} is an image composed of sub-pixels that correspond to I_{AA}. We adopt the structural similarity (SSIM) with a window size of 11×11 in the loss function and following sections. Parameters k and w are empirically set to 5 and 0.1, and $\phi(\cdot)$ is the last feature map of a deep network which is given by [51].

The training process of frame-recurrent based models usually takes a lot of time. Because the poor performance of each iteration can be transmitted along with the recurrent iterative process, especially at the beginning of training. To reduce the influence of this problem and accelerate network training, we recommend readers to use a pre-training trick that replaces the reused data I_{SS}^{t-1} with ground-truth image I_{GT}^{t-1} in the first several epochs. In our studies, it can help reduce the training time by hours while giving comparable and even better results.

4 Temporal Stability Assessment

In this section, we introduce a new video temporal stability metric termed inter-frame structural similarity (IF-SSIM) for rendered content. It evaluates the temporal stability by the differences of aligned consecutive frames rather than references. IF-SSIM takes a natural approach that checks the consistency of the temporal pixels in consecutive frames. We

(a) Aligned frames

(b) Masked and resampled frames

Fig. 5. Comparisons of *Frame t* and the warped *Frame t-1* using the ground-truth motion vectors. (a) is the raw frames and (b) is the processed frames in IF-SSIM evaluation.

obtain the flow of pixels by the renderer-generated motion vectors, and then use SSIM to evaluate the pixel-level temporal stability across frames. Although temporal pixels can be accurately aligned using rendered motion vectors, there are some critical issues that affect the evaluation of stability. First, not all of the pixels have their counterparts in previous frames, such as disoccluded pixels, as shown in Fig. 5 (a). Second, the color of a pixel may changes with time, like pixels on moving shades or non-Lambertian surfaces. Third, the interpolation operation in temporal pixel warping is equivalent to low-pass filtering, which can blur the warped frames. To tackle the above issues, we formulate IF-SSIM as masked inter-frame similarity based on human visual perception.

Given two consecutive frames, we first reconstruct the aligned two frames using a 2×2 pooling operation $\chi(\cdot)$ to counteract the effect of interpolating. Then, we need to remove the inherently mismatching pixels. The pixels of disoccluded regions can be easily found through depth tests. For the shading changed pixels, we distinguish them by comparing the max color variation of RGB channels with a perceptual difference threshold [52], [53]. We formulate the threshold as $T = f(B, D)$, *i.e.*, T only relates to the brightness B and screen type of the displayer D. A user experiment on different devices is conducted to determine $f(\cdot)$. Fig. 5 (b) shows the results after processing, which can be observed that the issues have been overcome. Lastly, the IF-SSIM value of a video with n frames is expressed as:

$$\text{IF-SSIM} = \frac{\sum_{t=2}^{n} \text{SSIM}\left(\chi(I_W^{t-1}) M_d M_p,\, \chi(I^t) M_d M_p\right)}{n-1}, \tag{4}$$

where I^t is *Frame t* and I_W^{t-1} is warped *Frame t-1*, the disoccluded mask M_d and the perceptual shading changed

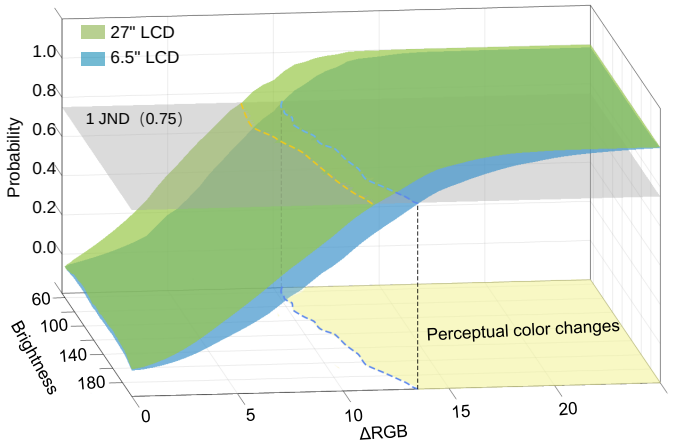


Fig. 6. The probability P of choosing the right changed object under different brightness and screens.

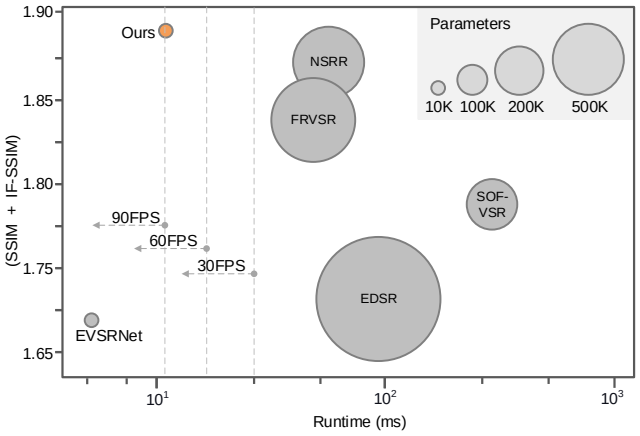


Fig. 7. Speed and performance comparison (NVIDIA 3090 GPU, PyTorch, and CUDA). Due to the lightweight design, our method can reach nearly 90 FPS. Moreover, it outperforms state-of-the-arts with high reconstruction quality.

mask M_p are computed using the depth and color maps:

$$\begin{cases} M_d = \{\, |\triangle \text{depth}| < 0.01 \,\} \\ M_p = \{\, max(|\triangle \text{RGB}|) < f(B,\, D) \,\}. \end{cases}$$

Note that the frames I^t and I_W^{t-1} are from the test videos, while the motion vectors and masks M_d and M_p are from the rendered ground-truth information.

To determine the perceptual difference threshold $T = f(B, D)$, we conduct a user experiment on the three game scenes (see Sec. 5). 20 participants, 10 males and 10 females with normal vision, are shown some short videos (60 frames) on the screens of PCs and mobile phones. We change the color of some small objects several times in the videos. Then the participants are asked whether they notice shading changes or flickers. We compute the probability P of choosing the right changed objects in the test. P of 75% is referred to as 1 just-noticeable-difference (JND) unit [54]. The modified objects below 1 JND can be considered as with no shading changes in human visual perception. After the test, we calculate the average brightness B of the color changed objects and their neighboring pixels (bounding box) and then given the probability P of perceptual color changes under different brightness B on a 27″ LCD computer display D_1 and a 6.5″ LCD mobile phone screen D_2, as shown in Fig. 6. The probabilities P rapidly increase with the growth of RGB color changes. The results on the two screens are approximate (difference less than 2 in ΔRGB). The curves of probability P reaching 1 JND, *i.e.*, the threshold $T = f(B, D)$, are given in the figure. We can further obtain the relation between the threshold T and the brightness B from the projected curve. For dark regions, the color changes are more likely to be noticed, because the human visual system is more sensitive to the differences between darker tones [55]. The RGB color changes on the right side of the curves are perceptible. In this paper, we evaluate the temporal stability of videos using the threshold $T_i = f(B_i, D_2)$ on the 6.5″ LCD mobile phone screen D_2 and using the average brightness of a 5×5 window of pixel i as its brightness B_i.

5 Datasets

We present a dataset called *GameVideo57* for training and fair comparisons of methods. It comprises 57 videos collected from three scenes, each video consists of 1,000 or 2,000 continuous frames.

Preparation. We build three scenes in Unity Engine, which represent three popular kinds of games, including multi-player online battle arena (MOBA) games, first-person view (FPV) games, and flight simulator (FS) games. To obtain the real players' data, three volunteers are employed to control the main character, and we record the moving paths. Then, 19 paths are selected for each game scene. We randomly choose 16 paths as the training set and the remaining 3 paths as the test set. After binding the camera to the main character, we capture the data for each path.

Data Generation. We develop dataset generation scripts to collect the training and test datasets using Unity Engine. It is necessary to disable some post-processing effects (e.g., bloom, motion blur, depth of field, and fog) that will produce visual artifacts in the results. These post-processing effects should be placed behind our method in the rendering pipeline. For reference images, we render each frame at resolution 2700×2700 with 8×MSAA and then downscale it to 900×900 using 3×3 average pooling. For LR input, we render input images at 300×300 for ×3 upscaling and at 450×450 for ×2 upscaling. Color and depth images are rendered with a jitter sequence in our method, as shown in Fig. 3, while motion vectors are rendered without jittering because motion vectors calculated at the center of pixel tiles are more stable. We also enable MSAA and apply the mip level bias approach where textures are sampled. The test data is also rendered by the same settings to evaluate the performances of different algorithms.

6 Experiments

6.1 Experimental Settings

We evaluate our proposed method on *GameVideo57* dataset by three metrics: peak signal-to-noise ratio (PSNR), structural similarity (SSIM) [56], and inter-frame structural similarity (IF-SSIM). PSNR and SSIM evaluate the similarity

SSIM	Scale	Bicubic	EDSR [10]	FRVSR [2]	SOF-VSR [3]	EVSRNet [11]	NSRR [4]	Ours	FRR-TAA
MOBA	×2	0.8501	0.8976	0.9182	0.9107	0.8825	0.9329	**0.9457**	0.9631
	×3	0.7592	0.8258	0.8783	0.8532	0.8346	0.9025	**0.9103**	
FPV	×2	0.8537	0.8704	0.9014	0.8892	0.8751	0.9317	**0.9381**	0.9609
	×3	0.7755	0.8055	0.8798	0.8183	0.7921	0.8914	**0.9023**	
FS	×2	0.8191	0.8396	0.8906	0.8451	0.8226	0.9012	**0.9196**	0.9553
	×3	0.7283	0.7607	0.8639	0.8051	0.7476	0.8736	**0.8853**	

PSNR	Scale	Bicubic	EDSR [10]	FRVSR [2]	SOF-VSR [3]	EVSRNet [11]	NSRR [4]	Ours	FRR-TAA
MOBA	×2	26.71	28.78	29.83	29.27	28.15	30.85	**32.31**	36.09
	×3	24.49	26.65	28.73	28.18	27.16	29.09	**29.66**	
FPV	×2	25.85	26.91	28.72	26.40	27.52	31.41	**32.10**	35.71
	×3	23.89	24.32	26.10	25.65	24.33	28.64	**29.17**	
FS	×2	25.34	26.03	28.41	26.93	26.34	30.21	**31.16**	34.23
	×3	23.20	24.44	26.53	24.97	24.06	27.52	**28.36**	

TABLE 1
Quantitative evaluation of generated images' quality on the three scenes for scale factor ×2 and ×3. The best performances (PSNR/SSIM) are shown in bold. As a reference, FRR-TAA represents full resolution rendering with temporal antialiasing.

IF-SSIM	Scale	Bicubic	EDSR [10]	FRVSR [2]	SOF-VSR [3]	EVSRNet [11]	NSRR [4]	Ours	FRR-TAA
MOBA	×2	0.9235	0.9465	0.9793	0.9627	0.9470	0.9832	**0.9847**	0.9912
	×3	0.9019	0.9204	0.9717	0.9585	0.9332	0.9814	**0.9817**	
FPV	×2	0.8949	0.9079	0.9714	0.9613	0.9009	0.9825	**0.9831**	0.9905
	×3	0.8562	0.8717	0.9685	0.9550	0.8619	**0.9807**	0.9803	
FS	×2	0.8941	0.9267	0.9713	0.9648	0.9341	0.9772	**0.9802**	0.9903
	×3	0.8671	0.8979	0.9651	0.9482	0.9179	0.9753	**0.9789**	

TABLE 2
Quantitative evaluation of temporal stability of generated videos. The best performances (IF-SSIM) are shown in bold.

between the reconstructed images and the ground-truth images, *i.e.*, image quality. IF-SSIM evaluates the pixel-level temporal stability among the generated frame sequences. In addition, we also give the running times of each module of our method on the personal computer and the mobile devices and compare it with other approaches. During the training phase, we use the LR videos as input data and the supersampled HR videos as output data. Details of the construction of the training and test datasets are addressed in Sec. 5. The training batch size is set to 6 for 50 epochs, and the test batch size is set to 1. For optimization, we use the Adam optimizer by setting $lr = 1e-3, betas = (0.9, 0.999)$, and $eps = 1e-4$. We present the results of ×2 and ×3 scaling approaches in Tabs. 1-2, respectively. Our experimental platforms are a desktop PC with an NVIDIA GeForce RTX 3090 GPU and two smartphones with Qualcomm Snapdragon 888 and 8 Gen1 SoCs.

Device	PC (1080p)	Smartphones–Snapdragon 888 (720p)	Smartphones–Snapdragon 8 Gen1 (720p)
Warping	0.19	0.21 (GPU)	0.19 (GPU)
Inpainting	0.43	0.73 (GPU)	0.58 (GPU)
Blending	11.56	8.95 (DSP)	7.79 (DSP)
Enhancement	0.25	0.96 (GPU)	0.84 (GPU)
Total	12.43 (ms)	10.85 (ms)	9.40 (ms)

TABLE 3
Running times of our ×3 upscaling models on PC (NVIDIA 3090 GPU) and smartphones (Qualcomm Snapdragon 888 and 8 Gen1 SoCs).

6.2 Comparisons

We compare our method with other approaches, including SISR method (EDSR [10]), VSR methods (FRVSR [2], SOF-VSR [3], and EVSRNet [11]), and the SOTA baseline NSRR [4]. We train and test all the methods on *GameVideo57* dataset with the same procedure. NSRR and our method take as input RGB images, depth images, and motion vectors. Different from other methods, the input of our method is rendered with alternate sub-pixel jittering sampling (see Sec. 3.2). It is worth pointing out that motion vectors and depth images are not used in SR methods, they usually use time-consuming optical flow estimation algorithms to obtain the motion information. To build a fair comparison between SR and neural supersampling approaches, we replace the optical flow with the accurate renderer-generated motion vectors for SR approaches.

Tab. 1 shows the comparisons of reconstructed image quality, and Tab. 2 shows the comparisons of the temporal stability of generated videos of different methods. The PSNR/SSIM and IF-SSIM values are given at different scenes and upscaling levels (×2 and ×3), where ×2 upscaling refers to shading 1 sample for each 4-pixels tile (*i.e.*, 25% shading budget), and ×3 refers to 11.1% shading budget. We also give the quantitative evaluation of full resolution rendered videos with temporal antialiasing (FRR-TAA), *i.e.*, 100% shading budget, as a reference. In general, the reconstruction results of learning-based methods are noticeably better than the bicubic interpolation method as

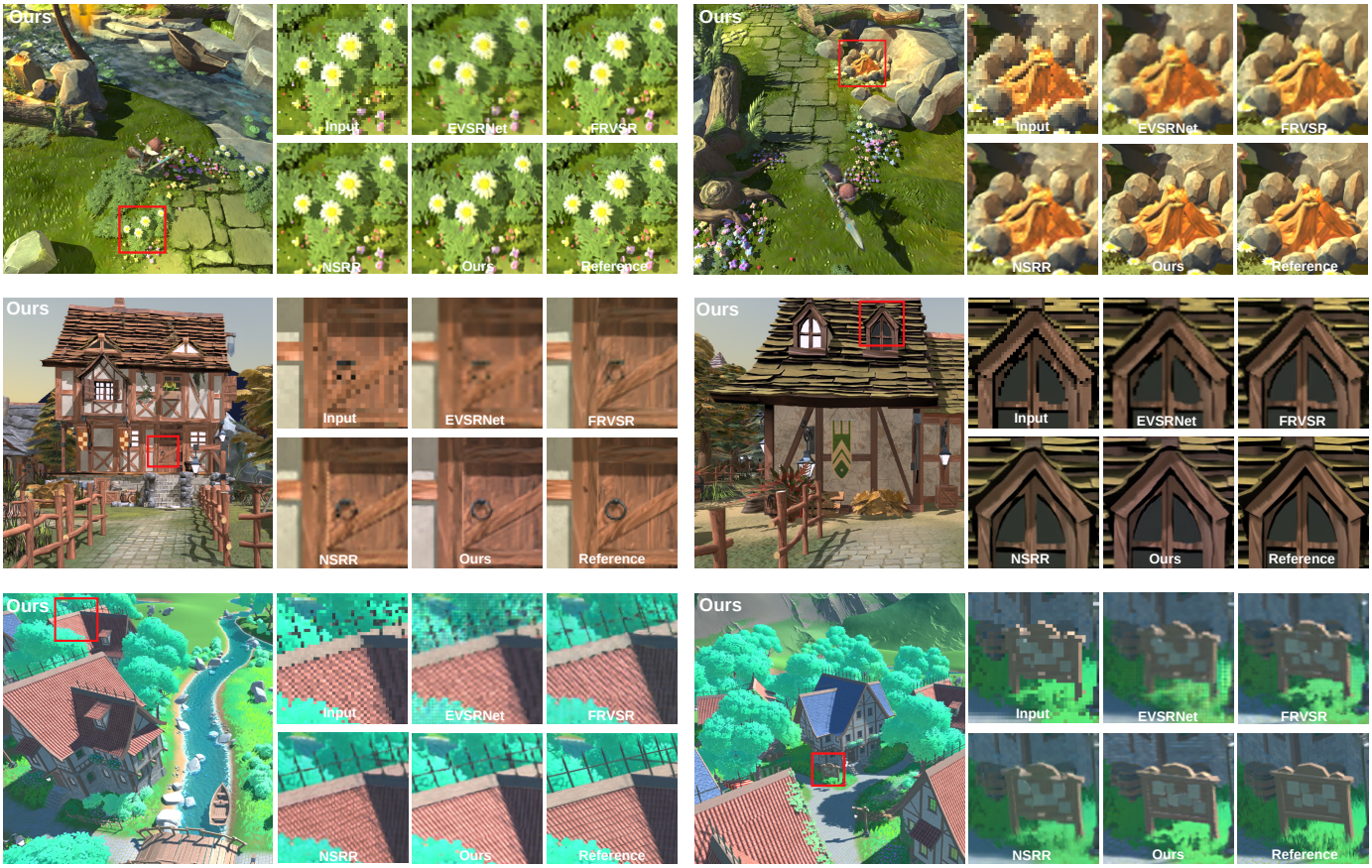


Fig. 8. Visual comparison results on the MOBA, FPV, and FS scenes for scale factor $\times 3$.

shown in Tabs. 1 and 2. Neural supersampling methods (NSRR and ours) outperform SISR and VSR approaches on both image quality (SSIM/PSNR) and temporal stability (IF-SSIM). Also, experiment results show that our method gives better results than the state-of-the-art supersampling method NSRR. Compared to the results of FRR-TAA which is rendered at the target resolution, our method achieves close SSIM scores for the MOBA and the FPV scenes. The gap between our method and FRR-TAA for the FS scene is relatively wider, as this scene contains much high frequency information. And we notice the superior temporal stability performance of TAA method in Tab. 2, which can also verify the effectiveness of the proposed metric IF-SSIM.

Fig. 8 shows visual comparisons of different methods on the three game scenes. We can see that SR approaches tend to produce smooth images that lack sufficient high-frequency details. The image quality of our method and NSRR are comparable, but our model is able to recover more fine details and gives more visually pleasing results. We also present a qualitative comparison of our method to two other solutions commonly used in PCs, DLSS 2.0 [7] and FSR 2.0 [24]. It should be noted that our training datasets were collected using the built-in render pipeline of the Unity Engine, which is widely used in mobile graphics applications. Our method relies on the forward shading features of Unity Engine's built-in pipeline, whereas DLSS 2.0 is currently only supported in Unity Engine's HDRP (High Definition Render Pipeline), and official support for FSR 2.0 in Unity has yet to be provided. As a result, we compare the performance of these three solutions qualitatively using results from different rendering engines and scenes. As shown in Fig. 9, the result of our method is rendered by the Unity Engine, while the results of DLSS 2.0 and FSR 2.0 are rendered by Unreal Engine 4, and all methods use a scaling factor of 2. We can see that our method and DLSS 2.0 provide clearer images with rich texture features for reconstruction image quality, whereas FSR 2.0 results are slightly blurry. Ghosting artifacts are barely visible in the results of DLSS and FSR methods for the disoccluded regions in the red boxes, with only a small blurry area appearing at the edge of moving objects. With fast moving objects, our method produces very slight ghosting, but it has little effect on video quality and temporal stability. We recommend that readers watch the demo video and compare the comparisons.

6.3 Model Performance Analysis

Real-time rendering applications usually run at more than 60 FPS, which poses a major challenge to the upscaling algorithms. In this section, we discuss the required computing resources and performances of the alternative super resolution and supersampling methods. As shown in Fig. 7, we give comparisons of parameters, running times, and performance on a personal computer. It is obvious that our method outperforms almost all baseline methods. Our model uses fewer running times and parameters while producing the best quality images. The video super resolution algorithm EVSRNet [11] is capable of working at high frame rates, but its reconstructed image quality is lower than that

Fig. 9. Visual comparisons of our method, DLSS 2.0 [7], and FSR 2.0 [24] for scale factor $\times 2$. The images in blue boxes compare different texture reconstruction methods, whereas the images in red boxes compare reconstructions on disoccluded regions.

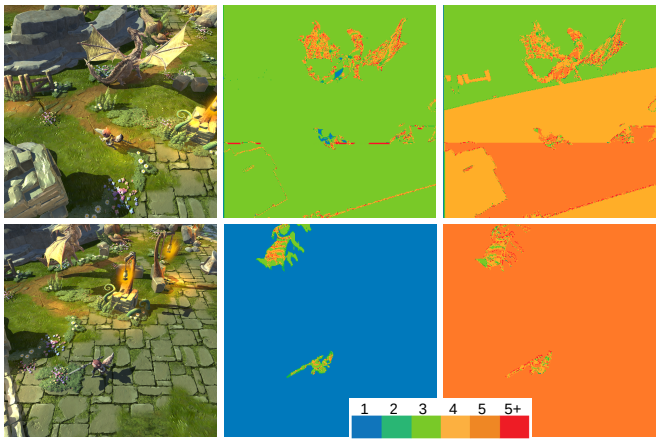


Fig. 10. Visual comparisons of the supersampling efficiency for $\times 3$ upscaling. For the given two images as shown in (a), where the character is moving fast in the top image, and stopped and is attacking in the bottom image. (b) and (c) are the visualized results of NSRR and our method. The colored pixels in (b) and (c) represent the numbers of sampled sub-pixels in 9 from temporal 5 frames. More sub-pixels sampled means higher supersampling efficiency.

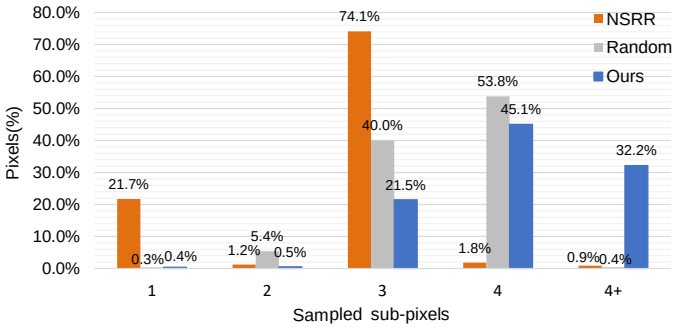


Fig. 11. The statistics of sampled sub-pixels of NSRR and our method on the test data for scale factor $\times 3$.

of other methods. Despite the time-consuming optical flow estimation networks of SR methods being replaced by free rendered motion vectors (as mentioned in Sec. 6.2), our method still achieves the best image quality at a high frame rate (reaching nearly 1080p & 90 FPS on NVIDIA GPU).

We then evaluate the performance of our model on the mobile devices. As discussed in Sec. 1 and Fig. 1, the neural supersampling model is trained on the collected rendered datasets on high performance computers before being deployed on mobile devices. The training is usually conducted using 32-bit floating-point arithmetic which allows for a large range of matrix operations. However, the trained 32-bit floating-point models are not well suited for use on mobile devices due to the expensive floating-point arithmetic. To reduce the model inference time and memory usage, we optimize our model using a post-training model quantization scheme [8]. It can quantize the 32-bit floating-point parameters of the trained network as 8-bit integers, which makes it applicable to run our model on mobile neural network accelerator hardware such as Qualcomm Hexagon digital signal processors (DSP). We run performance tests on smartphones powered by Qualcomm Snapdragon 888 and 8 Gen1 SoCs, respectively. The most computationally intensive model, the quantized reconstruction network, is loaded into the DSP units of the SoCs, while other simple modules, such as warping, inpainting, antialiasing, and enhancement networks, are performed on the GPU. Tab. 3 displays the test results. On smartphones, our model takes 10.85 and 9.40 milliseconds, respectively, with GPU processes taking 1.9 and 1.61 milliseconds and reconstruction taking 8.95 and 7.79 milliseconds. In theory, our model can achieve up to 90 FPS when the GPU and DSP work in tandem (when regardless of the memory copy cost, a discussion about this is placed in Sec. 6.6).

Adding the supersampling model to real-time rendering applications has little impact on the rendering performance on GPUs, since we perform most of the computations of our model on the low-power DSP units. In addition, the inference time and energy budget of our model will not change with the complexity of the rendering pipeline and 3D scenes. Therefore, using the supersampling model can achieve more benefits (*i.e.*, reducing the shading computational overhead while maintaining image quality) for complex scenarios.

6.4 Sampling Efficiency Analysis

In this section, we compare the sampling efficiency of the SOTA baseline NSRR [4], random subpixel sampling, and our sampling pattern. Based on the same temporal samples reusing approach, sampled sub-pixels of the previous 4 frames of NSRR and our method are warped to align with the current frame. Then we count the sampled sub-pixels of each pixel of LR images to show the sampling efficiency of different methods. More sampled sub-pixels means higher sampling efficiency and more image details can be recovered. Fig. 10 and 11 give the comparisons of sampled sub-pixels (up to 9 for $\times 3$ upscaling) of the methods on the test data. As shown in Fig. 10, given a fast-moving scene

Method	SSIM	PSNR	IF-SSIM
NSRR-Small	0.8631	27.57	0.9562
Ours	0.9103	29.66	0.9817
NSRR	0.9025	29.09	0.9814
Ours-Large	0.9197	29.74	0.9868

TABLE 4
Cross-validation experiment experiments for the reconstruction networks on the MOBA scene with scale factor $\times 3$. "NSRR-Small" and "Ours" adopt the proposed lightweight reconstruction network, while "NSRR" and "Ours-Large" adopt a heavy network used in [4]

Method	SSIM	PSNR	IF-SSIM
Center sampling	0.8816	28.09	**0.9829**
Random sampling	0.9033	29.11	0.9761
Ours	**0.9103**	**29.66**	0.9817

TABLE 5
Ablation experiments for jittering samples and the alignment module. We train our model with the different sampling settings (with "center", "random", and our sampling pattern) and report the results on the MOBA scene with scale factor $\times 3$.

(first row), NSRR gathers 3 sub-pixels (in green) for most pixels from 5 temporal frames, while our method gathers 3 to 5 sub-pixels (in green, orange, and orange-red). Given a static scene (second row), NSRR only gathers 1 sub-pixel (in blue) from 5 temporal frames, while our method gathers 5 sub-pixels (in orange-red). The illustrations in Fig. 10 are consistent with the results in Fig. 11. Our method can obtain 3 to 5 temporal sub-pixels for most pixels. NSRR only gathers 1 or 3 sampled sub-pixels for most pixels. Because much temporal samples are wasted (overlapped) at the same place, which limits its sampling efficiency.

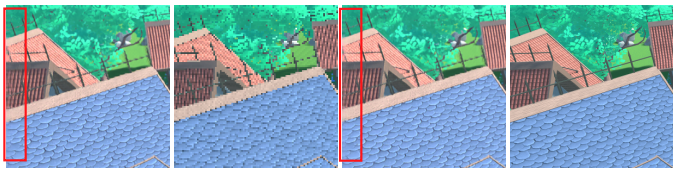
In addition, we execute a cross-validation experiment that swaps the reconstruction networks of NSRR and our method to verify the above analysis. As shown in Tab. 4, we replace the reconstruction network of NSRR with our lightweight network (NSRR Small) and adopt the reconstruction network of NSRR in our method (Ours-Large). Under the same network parameters, our supersampling framework gives better results with 0.0472, 2.09 dB, and 0.0255 improvements on SSIM, PSNR, and IF-SSIM for the light reconstruction network and 0.0172, 0.65 dB, and 0.0054 improvements for the large network compared to NSRR. The supersampling results NSRR relies on heavy reconstruction networks, which is not friendly to limited hardware, while our approach can generate stable results with lightweight neural networks.

6.5 Ablation Studies

Jittering Samples. In Sec. 3.2 we propose to use a new sample pattern for the pixel shading process. Tab. 5 shows an ablation study on the effectiveness of different sampling patterns. To effectively collect temporal samples, we apply sub-pixel jitter to the input image to produce uniformly distributed sub-pixel samples. By performing jittering sample pattern, the output image quality is significantly improved from SSIM 0.8816 to 0.9103 and PSNR 28.09 to 29.66 dB.

Method	SSIM	PSNR	IF-SSIM
Single frame	0.8463	27.56	0.9412
Reuse I_L^{t-1}	0.8625	28.28	0.9584
Reuse I_{SS}^{t-1} (Ours)	**0.9103**	**29.66**	**0.9817**

TABLE 6
Ablation experiments for recurrently reusing a history frame. We train our model with the settings corresponding to the first column in the table and report the results on the MOBA scene with scale factor $\times 3$.



(a) History frame (b) Input (c) Ours (d) Reference

Fig. 12. A failure case. When the camera moves fast to the left, there will be noticeable artifacts on the left edge of the result, and the error will stay for several frames.

Our sampling pattern also achieves better results compared to random sampling. Note that, the IF-SSIM with a jittering sampling pattern is slightly lower than the baseline of center sampling, caused by the small inconsistency introduced by the jittered samples between consecutive frames, which leads to slightly worse temporal stability. However, this is a reasonable compromise given the 0.0287 and 1.57 dB improvement on SSIM and PSNR at the cost of merely 0.0012 degradation on IF-SSIM.

Reuse History Frame. Tab. 6 shows the results of three strategies. As the single-frame approach cannot leverage temporal information, its overall performance is unsurprisingly the worst. Next, reusing the previous frame which does not carry network-generated information cannot make full use of temporal samples efficiently. Therefore, the approach of reusing I_L^{t-1} results in image quality and temporal stability degradation. Finally, our full model achieves the best results with 0.0640, 2.10 dB, and 0.0450 improvements on SSIM, PSNR, and IF-SSIM respectively.

6.6 Limitations and Future Work

Our method has some limitations. First, our method depends on the quality of the history frame. When significant changes happen, the history frame may not contain sufficient information. In this case, errors will arise (Fig. 12), but they can be repaired by the following samples quickly. Second, although our method outperforms SOTAs in runtime performance, it can still be demanding for lightweight hardware such as low-end smartphones. We believe that with the architectural upgrades of mobile SoCs and more powerful AI units embedded, the speed can be greatly accelerated, and the power consumption can be notably reduced. Third, AI units typically have their own memory subsystem, and when offloading real-time tasks from the GPU to the AI unit, the cost of memory transfer cannot be overlooked. Despite the fact that we use ION buffers [57] to map memory from the GPU to the AI unit and the FastRPC protocol [58] to facilitate remote procedure calls between the

CPU and AI unit, the memory transfer process on the test SoC still takes about 0.8∼3ms, resulting in a small additional latency.

Our method can be improved in the following directions. (1) Our supersampling framework is naturally applicable to checkerboard rendering and variable rate shading. If the hardware supports these new techniques, integrating the neural supersampling model into the rendering pipeline can produce better quality images. (2) And we are also interested in exploiting more auxiliary parameters available in modern renderers such as object ID to recover the edges of objects better.

7 Conclusion

Given the success of neural supersampling for real-time rendering on high-end computers, we present an efficient neural supersampling framework for real-time rendering applications on smartphones. To maximize the sampling efficiency, we propose a sub-pixel sample pattern to gather more spatio-temporal information at the limited sampling rates. To deal with the challenges of computation constraints on the mobile platform, our method uses an efficient inpainting module to rectify invalid history pixels and adopts lightweight networks and quantization techniques to reduce the computational cost of reconstruction networks. It achieves high image reconstruction quality while keeping temporal stability by using a recurrent-frame approach. We also demonstrate the superior runtime performance of our method. To our best knowledge, we present the first real-time neural supersampling method on smartphones, which may benefit various real-time applications. We test our model on a moderate mobile platform, and the performance reaches up to 90 FPS for 720p images. Finally, we introduce a new metric termed IF-SSIM to measure pixel-level temporal stability quantitatively, and a new dataset called *GameVideo57* to the lack of data for future research.

Acknowledgments

This work was supported by Key R&D Program of Zhejiang (No. 2023C01047), and the National Natural Science Foundation of China (Grant Nos. 62036010, 61972344).

References

[1] J. Guo, X. Fu, L. Lin, H. Ma, Y. Guo, S. Liu, and L.-Q. Yan, "Extranet: real-time extrapolated rendering for low-latency temporal supersampling," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 6, pp. 1–16, 2021.

[2] M. S. Sajjadi, R. Vemulapalli, and M. Brown, "Frame-recurrent video super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6626–6634.

[3] L. Wang, Y. Guo, Z. Lin, X. Deng, and W. An, "Learning for video super-resolution through hr optical flow estimation," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 514–529.

[4] L. Xiao, S. Nouri, M. Chapman, A. Fix, D. Lanman, and A. Kaplanyan, "Neural supersampling for real-time rendering," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 142–1–12, 2020.

[5] X. Wei, H. Huang, Y. Shi, H. Yuan, L. Shen, and J. Wang, "End-to-end adaptive monte carlo denoising and super-resolution," *arXiv preprint arXiv:2108.06915*, 2021.

[6] A. Edelsten, P. Jukarainen, and A. Patney, "Truly next-gen: Adding deep learning to games and graphics," in *Game Developers Conference*, 2019.

[7] E. Liu, "Dlss 2.0-image reconstruction for real-time rendering with deep learning," in *GPU Technology Conference (GTC)*, 2020.

[8] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[9] U. Technologies, "Unity engine," http://unity3d.com, 2005-2021.

[10] 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.

[11] S. Liu, C. Zheng, K. Lu, S. Gao, N. Wang, B. Wang, D. Zhang, X. Zhang, and T. Xu, "Evsrnet: Efficient video super-resolution with neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2480–2485.

[12] J. D. Hobby, "Rasterization of nonparametric curves," *ACM Trans. Graph.*, vol. 9, no. 3, p. 262–277, 1990. [Online]. Available: https://doi.org/10.1145/78964.78966

[13] K. Akeley, "Reality engine graphics," in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '93, New York, NY, USA, 1993, p. 109–116. [Online]. Available: https://doi.org/10.1145/166117.166131

[14] J. Jimenez, J. I. Echevarria, T. Sousa, and D. Gutierrez, "Smaa: enhanced subpixel morphological antialiasing," in *Computer Graphics Forum*, vol. 31, no. 2pt1, 2012, pp. 355–364.

[15] B. Karis, "High quality temporal supersampling," in *Advances in Real-Time Rendering for Games, SIGGRAPH Courses*, 2014.

[16] N. Damera-Venkata and N. L. Chang, "Display supersampling," *ACM Trans. Graph.*, vol. 28, no. 1, 2009. [Online]. Available: https://doi.org/10.1145/1477926.1477935

[17] L. Yang, D. Nehab, P. V. Sander, P. Sitthi-amorn, J. Lawrence, and H. Hoppe, "Amortized supersampling," in *ACM SIGGRAPH Asia 2009 Papers*, ser. SIGGRAPH Asia '09, 2009.

[18] L. Yang, S. Liu, and M. Salvi, "A survey of temporal antialiasing techniques," in *Computer Graphics Forum*, vol. 39, no. 2, 2020, pp. 607–621.

[19] K. Vaidyanathan, M. Salvi, R. Toth, T. Foley, T. Akenine-Möller, J. Nilsson, J. Munkberg, J. Hasselgren, M. Sugihara, P. Clarberg, T. Janczak, and A. Lefohn, "Coarse pixel shading," in *Proceedings of High Performance Graphics*, ser. HPG '14, 2014, p. 9–18.

[20] J. E. El Mansouri, "Rendering 'rainbow six—siege'," in *Game Developers Conference*, 2016.

[21] E. Games, "Unreal engine 4.27: Screen percentage with temporal upsample," https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/ScreenPercentage/, 2021.

[22] M. M. Thomas, K. Vaidyanathan, G. Liktor, and A. G. Forbes, "A reduced-precision network for image reconstruction," *ACM Trans. Graph.*, vol. 39, no. 6, Nov. 2020. [Online]. Available: https://doi.org/10.1145/3414685.3417786

[23] AMD, "Fidelityfx super resolution," https://www.amd.com/en/technologies/radeon-software-fidelityfx-super-resolution, 2021.

[24] ——, "Fidelityfx super resolution 2.0," https://gpuopen.com/fidelityfx-superresolution-2/, 2022.

[25] 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.

[26] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1646–1654.

[27] X. Yang, H. Mei, J. Zhang, K. Xu, B. Yin, Q. Zhang, and X. Wei, "Drfn: Deep recurrent fusion network for single-image super-resolution with large factors," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 328–337, 2018.

[28] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2472–2481.

[29] J. Caballero, C. Ledig, A. Aitken, A. Acosta, J. Totz, Z. Wang, and W. Shi, "Real-time video super-resolution with spatio-temporal networks and motion compensation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4778–4787.

[30] K. C. Chan, X. Wang, K. Yu, C. Dong, and C. C. Loy, "Basicvsr: The search for essential components in video super-resolution and beyond," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4947–4956.
[31] Z. Wang, J. Chen, and S. C. Hoi, "Deep learning for image super-resolution: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2020.
[32] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3147–3155.
[33] H. Dou, C. Chen, X. Hu, Z. Xuan, Z. Hu, and S. Peng, "Pca-srgan: Incremental orthogonal projection discrimination for face super-resolution," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1891–1899.
[34] H. Liu, Z. Ruan, P. Zhao, C. Dong, F. Shang, Y. Liu, and L. Yang, "Video super resolution based on deep learning: A comprehensive survey," *arXiv preprint arXiv:2007.12928*, 2020.
[35] X. Wang, K. C. Chan, K. Yu, C. Dong, and C. Change Loy, "Edvr: Video restoration with enhanced deformable convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.
[36] X. Ying, L. Wang, Y. Wang, W. Sheng, W. An, and Y. Guo, "Deformable 3d convolution for video super-resolution," *IEEE Signal Processing Letters*, vol. 27, pp. 1500–1504, 2020.
[37] X. Tao, H. Gao, R. Liao, J. Wang, and J. Jia, "Detail-revealing deep video super-resolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4472–4480.
[38] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.
[39] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 764–773.
[40] S. Y. Kim, J. Lim, T. Na, and M. Kim, "Video super-resolution based on 3d-cnns with consideration of scene change," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 2831–2835.
[41] Y. Huang, W. Wang, and L. Wang, "Bidirectional recurrent convolutional networks for multi-frame super-resolution," *Advances in neural information processing systems*, vol. 28, pp. 235–243, 2015.
[42] J. Guo and H. Chao, "Building an end-to-end spatial-temporal convolutional network for video super-resolution," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
[43] R. Soundararajan and A. C. Bovik, "Video quality assessment by reduced reference spatio-temporal entropic differencing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 4, pp. 684–694, 2012.
[44] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "Toward a practical perceptual video quality metric," *The Netflix Tech Blog*, vol. 6, no. 2, 2016.
[45] P. G. Barten, "Formula for the contrast sensitivity of the human eye," in *Image Quality and System Performance*, vol. 5294. SPIE, 2003, pp. 231–238.
[46] T. O. Aydin, M. Čadík, K. Myszkowski, and H.-P. Seidel, "Video quality assessment for computer graphics applications," *Acm Transactions on Graphics (Tog)*, vol. 29, no. 6, pp. 1–12, 2010.
[47] R. K. Mantiuk, G. Denes, A. Chapiro, A. Kaplanyan, G. Rufo, R. Bachy, T. Lian, and A. Patney, "Fovvideovdp: A visible difference predictor for wide field-of-view video," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–19, 2021.
[48] D. Fuoli, S. Gu, and R. Timofte, "Efficient video super-resolution through recurrent latent space propagation," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 2019, pp. 3476–3485.
[49] Z. Zeng, S. Liu, J. Yang, L. Wang, and L.-Q. Yan, "Temporally reliable motion vectors for real-time ray tracing," in *Computer Graphics Forum*, vol. 40, no. 2. Wiley Online Library, 2021, pp. 79–90.
[50] A. Polesel, G. Ramponi, and V. J. Mathews, "Image enhancement via adaptive unsharp masking," *IEEE transactions on image processing*, vol. 9, no. 3, pp. 505–510, 2000.
[51] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.
[52] H. R. Blackwell, "Luminance difference thresholds," in *Visual psychophysics*. Springer, 1972, pp. 78–101.
[53] J. H. Mueller, T. Neff, P. Voglreiter, M. Steinberger, and D. Schmalstieg, "Temporally adaptive shading reuse for real-time rendering and virtual reality," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 2, pp. 1–14, 2021.
[54] R. K. Mantiuk, A. Tomaszewska, and R. Mantiuk, "Comparison of four subjective methods for image quality assessment," in *Computer graphics forum*, vol. 31, no. 8. Wiley Online Library, 2012, pp. 2478–2491.
[55] C. Poynton, *Digital video and HD: Algorithms and Interfaces*. Elsevier, 2012.
[56] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.
[57] T. M. Zeng, "The android ion memory allocator," *Linux Weekly News*, 2012.
[58] M. Deore, "Hexagon dsp — cpu offload," https://mdeore.medium.com/hexagon-dsp-cpu-offload-4fb8e4077fe8, 2017.

Sipeng Yang received the BSc degree from Southeast University, P.R. China, in 2018. He is currently a PhD candidate in the State Key Lab of CAD&CG, Zhejiang University, China. His research interests include computer graphics and machine learning.

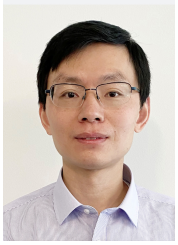
Yunlu Zhao received the BSc degree in Computer Science and Technology from Hangzhou Dianzi University, China, in 2018. He is currently pursuing an MSc degree in Software Engineering at the State Key Lab of CAD&CG, Zhejiang University, China. His research interests are mainly in real-time rendering.

Yuzhe Luo received the BSc degree from Zhejiang University, P.R. China, in 2022. He is currently a graduate student in the State Key Lab of CAD&CG, Zhejiang University, China.

He Wang is an Associate Professor at the School of Computing, University of Leeds, UK. He is a Turing Fellow, the Director of High-Performance Graphics and Game Engineering and Academic Lead at the Centre for Immersive Technology at Leeds. His research interest is mainly in computer graphics, computer vision and machine learning. Previously he was a Senior Research Associate at Disney Research Los Angeles. He received his PhD from the University of Edinburgh, UK.

Hongyu Sun is head of computing graphics research institute in OPPO, responsible for converting state of the art graphics technologies into products. He is particularly focused on efficient and realistic rendering, both traditional and AI-powered. Before OPPO, he was chief software architect for Huawei, terminal OS dept, responsible for graphics and computer vision features. He holds a PhD from Iowa State University, where he worked under Dr. Robyn Lutz.

Chen Li received B.S. and M.S. degree in Computer Science from University of Science and Technology of China in year 1999 and 2002 respectively. He joined Microsoft Research in year 2004. He moved to U.S. in year 2008 and joined Microsoft Xbox Team. He joined Facebook Reality Lab in year 2017, worked on various AR/VR projects. He joined OPPO's Software Engineering Institute in year 2019. He's currently the head of OPPO's graphics research engineering team located in Bellevue, WA.



Binghuang Cai received the B.S. degree in electronic information engineering and the M.S. degree in signal and information processing from Shantou University, Shantou, China, in 2004 and 2007, respectively, and the Ph.D. degree in communication and information systems from Sun Yat-sen University, Guangzhou, China, in 2010. He became a Research Fellow at University of Technology, Sydney, Australia, in 2010; a Postdoctoral Associate at the University of Pittsburgh, Pittsburgh, PA, in 2012; a Senior Fellow at the University of Washington, Seattle, WA, in 2015; and a Scientist at Allen Institute for Brain Science, Seattle, WA, in 2017. Dr. Cai is currently a Researcher at OPPO US Research Center, Seattle, WA. His research interests include AI, computer graphics, computer vision, robotics, brain science and biomedical informatics.

Xiaogang Jin received a BSc degree, an MSc degree, and a PhD degree from Zhejiang University, P.R. China, in 1989, 1992, and 1995, respectively. He is a professor in the State Key Laboratory of CAD&CG, Zhejiang University. His current research interests include traffic simulation, collective behavior simulation, cloth animation, virtual try-on, digital face, implicit surface modeling and applications, creative modeling, computer-generated marbling, sketch-based modeling, and virtual reality. He received an ACM Recognition of Service Award in 2015 and the Best Paper Awards from CASA 2017 and CASA 2018. He is a member of the IEEE and the ACM.