Taming High-Resolution Auxiliary G-Buffers for Deep Supersampling of Rendered Content

Pengjie Wang, Chengzhi Yuan, Jie Guo, Xiaosong Yang, Houjie Li, Ian Stephenson, Jian Chang, and Ying Cao

***Abstract*—High-resolution images come with rich color information and texture details. Due to the rapid upgrading of display devices and rendering technologies, high-resolution real-time rendering faces the computational overhead challenge. To address this, the current mainstream solution is to render at a lower resolution and then upsample to the target resolution by supersampling techniques. However, while many prior supersampling approaches have attempted to exploit rich rendered data such as color, depth, motion vectors at low resolution, there is little discussion on how to harness high-frequency information that is readily available in the high-resolution (HR) G-buffers of modern renders. In this article, we seek to investigate how to *fully* leverage information from HR G-buffers to maximize the visual quality of supersampling results. We propose a neural network for real-time supersampling of rendered content, which is based on several core designs, including gated G-buffers encoder, G-buffers attended encoder and reflection-aware loss. These designs are especially made for the sake of effectively using HR G-buffers, enabling faithful recovery of a variety of high-frequency scene details from low-resolution, highly aliased inputs. Furthermore, a simple occlusion-aware blender is proposed to efficiently rectify dis-occluded features in the warped previous frame, allowing us to better exploit history information to improve temporal stability. The experiments show that our method, equipped with strong ability to harness HR G-buffer information, significantly improves the visual fidelity of high-resolution reconstructions upon previous state-of-the-art methods, even for challenging 4×4 upsampling, while still being compute-efficient.**

***Index Terms*—Rendering, supersampling, upscaling.**

Received 27 December 2024; revised 12 August 2025; accepted 10 September 2025. Date of publication 12 September 2025; date of current version 10 November 2025. This work was supported in part by the European Union's Horizon 2020 research and innovation programme (Ref.: Marie Skłodowska-Curie) under Grant 900025 and in part by the Liaoning Provincial Education Department, China under Grant LJ242412026018. Recommended for acceptance by Y. Dong. *(Corresponding author: Ying Cao.)*

Pengjie Wang is with the School of Computer Science, Dalian Minzu University, Dalian 116620, China, and also with the National Center for Computer Animation, Bournemouth University, BH12 5BB Poole, U.K. (e-mail: pengjiewang@gmail.com).

Chengzhi Yuan and Houjie Li are with the School of Information and Communication Engineering, Dalian Minzu University, Dalian 116620, China (e-mail: orangecircle128@gmail.com; lhj@dlnu.edu.cn).

Jie Guo is with the Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China (e-mail: guojie@nju.edu.cn).

Xiaosong Yang, Ian Stephenson, and Jian Chang are with the National Center for Computer Animation, Bournemouth University, BH12 5BB Poole, U.K. (e-mail: xyang@bournemouth.ac.uk; istephen@bournemouth.ac.uk; jchang@bournemouth.ac.uk).

Ying Cao is with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China (e-mail: caoying59@gmail.com).

Digital Object Identifier 10.1109/TVCG.2025.3609456

I. Introduction

HIGH-QUALITY real-time rendering is essential in 3D industries, including film, television production and video games. However, as 3D models and textures become more detailed and realistic, and as advanced rendering techniques such as ray tracing and global illumination are widely adopted, the cost of rendering high-resolution 3D scenes escalates significantly. This presents a challenge for real-time rendering. Consequently, consumers often find themselves having to balance between image quality and frame rate.

To tackle this problem, there is a rising research interest on neural supersampling in recent years, and some promising techniques such as DLSS [1], FSR [2], and XeSS [3] have been proposed. These methods render the current frame at low-resolution (LR) to reduce the computational cost, and train neural networks to combine the past frames to reconstruct the high-resolution (HR) image. However, while these methods can obtain satisfactory quality for 2×2 supersampling, their performance degrades by large margin when dealing with 4×4 supersampling. NSRR [4] achieves high-resolution image reconstruction for 4×4 supersampling, but it comes with high computational cost and memory requirement, and the reconstruction quality on complex scenes is still unsatisfactory. MNSS [5] presents a lightweight 4×4 supersampling approach, but it comes at the cost of reconstruction quality. Balancing image quality and efficiency remains a key challenge in supersampling research.

To further improve reconstruction quality, one of the possible avenues is to seek for more auxiliary input information, particularly on high-frequency details, which can be beneficial to the supersampling task. Some Monte Carlo denoising [6], [7] and video frame extrapolation works [8], [9] have explored geometric buffer (G-buffer) as auxiliary information, which contains pixel-based detailed representations of a scene (e.g., base color, depth, normal). Since the G-buffer is generated before the lighting pass, the computational overhead is relatively small even for the high-resolution G-buffer. A recent study [10] has shown the effectiveness of HR G-buffers as auxiliary information for high-fidelity upsampling. However, this method suffers from inability to fully utilize the G-buffers information due to the use of simple downsampling and fusion strategies, which limits its performance on scenes with complex textures.

In this paper, we propose a novel method for real-time supersampling of rendered content which considers both rendering attributes (e.g., color, depth and motion vectors) for low-resolution frames and HR G-buffers, and the high-resolution image of the

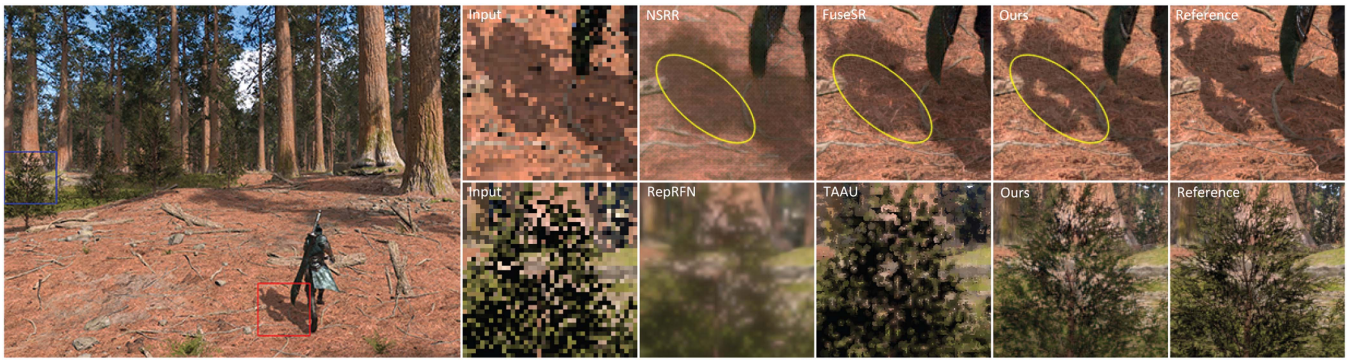


Fig. 1. Real-time 4×4 supersampling results of our method on the Redwood Forest scene. By leveraging high-frequency information from high-resolution G-buffers and fusing visual features from history frames in a occlusion-aware manner, our method significantly improves the reconstruction quality in various scenarios, especially those with complex textures, compared to existing methods.

target frame is reconstructed by fusing their features. Different from [10], our method can fully make use of information of HR G-buffers through several simple yet effective architecture and loss designs. In particular, we propose two novel components and a novel loss function, *gated G-buffers encoder*, *G-buffers attended encoder* and *reflection-aware loss*. Our first component addresses different G-buffers unequally by introducing base-color and depth maps guided gate fusion network to extract scene detail information while avoiding redundant information. The second component employs gradient map derived from base-color and normal maps to enhance the current frame features. Last but not least, we propose a *reflection-aware loss* that applies pixel-wise weighting to let our network focus on regions constrained by masks computed from normal, metallic and roughness maps of G-buffers to optimize the reconstruction of highly reflective areas.

In addition, to guarantee the temporal coherence of results, similar to prior works [4], [5], [10], our method is conditioned on low-resolution history frames by warping them towards the current frame using rendered motion vectors. However, we notice a limitation with these approaches: they learn to re-weight the features of the warped previous frames to address invalid features caused by dynamic dis-occlusion, which may result in the loss of information of valid features. To address the above problem, we propose a simple and efficient *occlusion-aware blender*, which directly identifies invalid features based on depth and color comparison. These regions are then masked and image inpainting is conducted for the holes under the fusion of various auxiliary features, thereby obtaining accurate history frame features.

Finally, we introduce a fusion network based on UNet [11] to realize deep fusion of G-buffer features with frame features, guiding the reconstruction of texture details. With the help of these new components, our method provides a more faithful reconstruction of complex, fine-grained scene details compared to existing methods, as shown in Fig. 1.

Our main contributions are:

- We propose a supersampling neural network for real-time rendering, which is capable of faithfully reconstructing the high-frequency details of underlying scenes from highly aliased inputs.
- We propose fully making use of the features from G-buffers by introducing *gated G-buffers encoder*, *G-buffers attended encoder* and *reflection-aware loss*, which can employ specific G-buffer maps to efficiently guide effective feature fusion and preserve high-frequency details. An additional *occlusion-aware blender* is introduced to improve the quality of history frame features for better temporal coherence.
- Experiments on several challenging scenes with large motion dynamics and complex texture details show that our model can produce significantly better results in terms of spatial and temporal fidelity, compared to existing state-of-the-art methods, with less computational demands. Furthermore, our method can achieve performance competitive with or better than popular commercial techniques such as DLSS and FSR.

II. Related Work

A. Antialiasing

In real-time rendering, the resolution increase of the rendered image is similar to the step-by-step processing of image details, which is accomplished by point sampling of the rendered image for each pixel. The use of traditional sampling methods in graphics (e.g., point sampling) results in jagged edges in the image. Antialiasing techniques such as multisampling (MSAA) [12] and temporal antialiasing (TAA) [13], [14] can improve image quality by increasing the sampling rate or sampling the image multiple times, but they also increase rendering costs. FSR [2] and TAAU [15] are based on traditional image processing techniques, like motion estimation and sharpening, utilizing temporal anti-aliasing and reconstruction methods to enhance image quality by leveraging information from multiple frames. They have achieved significant success in the gaming industry. However, they can still encounter aliasing issues in highly dynamic or complex scenes.

Deep learning-based supersampling techniques have received much attention in recent years. Deep Learning Super Sampling (DLSS) [1] uses neural networks to enhance low-resolution frames, dramatically lowering the rendering cost of high-resolution frames and enabling real-time rendering with

superior quality. In addition to DLSS there are a number of deep learning based methods such as XeSS [3]. However, most of these methods can only achieve favorable reconstruction quality with upsampling factor smaller than 2×2, and DLSS also requires specific hardware support. NSRR [4] recursively backward warps multiple history frames and depth maps with motion vectors to deliver impressive outcomes in demanding 4×4 upsampling tasks. Nevertheless, the approach has certain constraints in precisely restoring high-resolution details, and its inference is slower since it works at high resolution. DFASR [16] leverages frequency-domain similarity to improve the integration of geometric features, which is the first to propose an arbitrary-scale rendering super-resolution approach, capable of generating high-quality results at multiple magnification levels using a single model. NSRD [17] decomposes the rendered image into illumination and material components, focusing super-resolution efforts solely on the illumination part. This decomposition simplifies the learning objective and reduces reconstruction complexity. To further improve temporal consistency, it introduces a reliable warping mechanism along with a frame-recurrent strategy. MNSS [5] designs a lightweight superpampling framework with a sub-pixel sampling pattern, which accumulates history frames into a buffer and adopts a streamlined network design to attain competitive runtime efficiency. The most closely related work to ours is FuseSR[10], which addresses the challenging 8×8 super-resolution task by introducing HR G-buffers into its pipeline, resulting in high-quality outputs. However, FuseSR uses only simple downsampling and fusion strategies, which do not fully utilize the HR G-buffers information. In contrast, we specifically design a gated G-buffers encoder to effectively extract high-frequency features from HR G-buffers and utilize a G-buffers attended encoder to enhance the edges of the frame features, enabling our model to better exploit the HR G-buffers information.

B. Super-Resolution

Single-frame super-resolution. Conventional image super-resolution methods, while simple and fast-running, are limited in their effectiveness in recovering fine details. In contrast, deep learning-based methods have achieved notable advancements in the area of image super-resolution. These methods use neural networks to learn high-level representations of images that can better reconstruct lost details and produce more realistic and sharp images. SRCNN [18] represents the pioneering effort in applying deep learning to super-resolution reconstruction. EDSR [19] further improves the reconstruction quality by utilizing more residual blocks. RDN [20] incorporates the residual dense block to extract the more effective hierarchical features. RCAN [21] introduces an attention mechanism to distinguish the features of different channels and proposes a Residual in Residual structure. RepRFN [22] is designed with a multi-branch structure to capture and fuse as many features as possible from different modalities and a reparameterization operation is introduced to enable the utilization of the complex multi-branch structure even in lightweight networks. CAMixer [23] employs a trainable predictor to generate several bootstraps, which modulates attention to incorporate more beneficial textures in a self-adaptive manner, thereby enhancing the convolution's representation ability. However, these methods struggle to achieve good super-resolution results in real-time rendering tasks. They are typically more suited for low-resolution camera imagery, which suffers from blurring caused by integration rather than aliasing.

Multi-frame super-resolution. Super-resolution methods based on multi-frame images are commonly used in the video field. They utilize time-domain information and consider the correlation of neighboring image frames to enhance the performance of super-resolution results. This approach makes full use of inter-frame sharing information and is particularly suitable for video compression, video enhancement and HD video reconstruction. VESPCN [24] uses motion compensation in conjunction with video super-resolution to enhance performance. EDVR [25] employs a spatio-temporal attention fusion module and a cascade deformable alignment module. DUF [26] utilizes dynamic upsampling filters to avoid explicit motion compensation, improving video quality and temporal consistency. MoTIF [27] uses spatio-temporal implicit neural functions and reliable splatting and decoding schemes to fuse spatio-temporal information from various reference frames. TCNet [28] explores temporal consistency by employing the correlation matching and an adaptive warping to maintain structural stability. However, most of these methods rely on both history and future frames to reconstruct the current frame, making them difficult to apply in supersampling, where typically only history frames are available.

C. Image Generation and Restoration With G-Buffers

In real-time rendering, G-buffer acts as an important technology to save and process key information in the scene, enabling more realistic and efficient graphics rendering in the subsequent stages. Researchers have attempted to employ rich information in G-buffers for various image generation and restoration tasks. For example, ExtraNet [8] uses the albedo, depth, normal, roughness, and metallic information in G-buffers and history frames for frame extrapolation prediction. This approach produces reasonable extrapolated results without introducing significant artifacts and has been successful in improving frame rates in practice. Zhang et al. [7] introduce a Monte Carlo denoising network that leverages subpixel sampling patterns to restore high-quality image sequences while maintaining real-time frame rates, delivering superior denoising results. Ioannou et al. [29] leverages G-buffer information to achieve in-game artistic stylization while preserving temporally consistent and scene details.

Additionally, G-buffers play a critical role in some studies that address supersampling in conjunction with other problems. Wei et al. [30] combines Monte Carlo denoising and supersampling, which applies G-buffer features to conditional feature modulation at each layer to reconstruct scene details. ExtraSS [9] addresses both frame generation and supersampling, using G-buffer-guided warping to handle motion shadows, thereby obtaining more accurate spatiotemporal information.

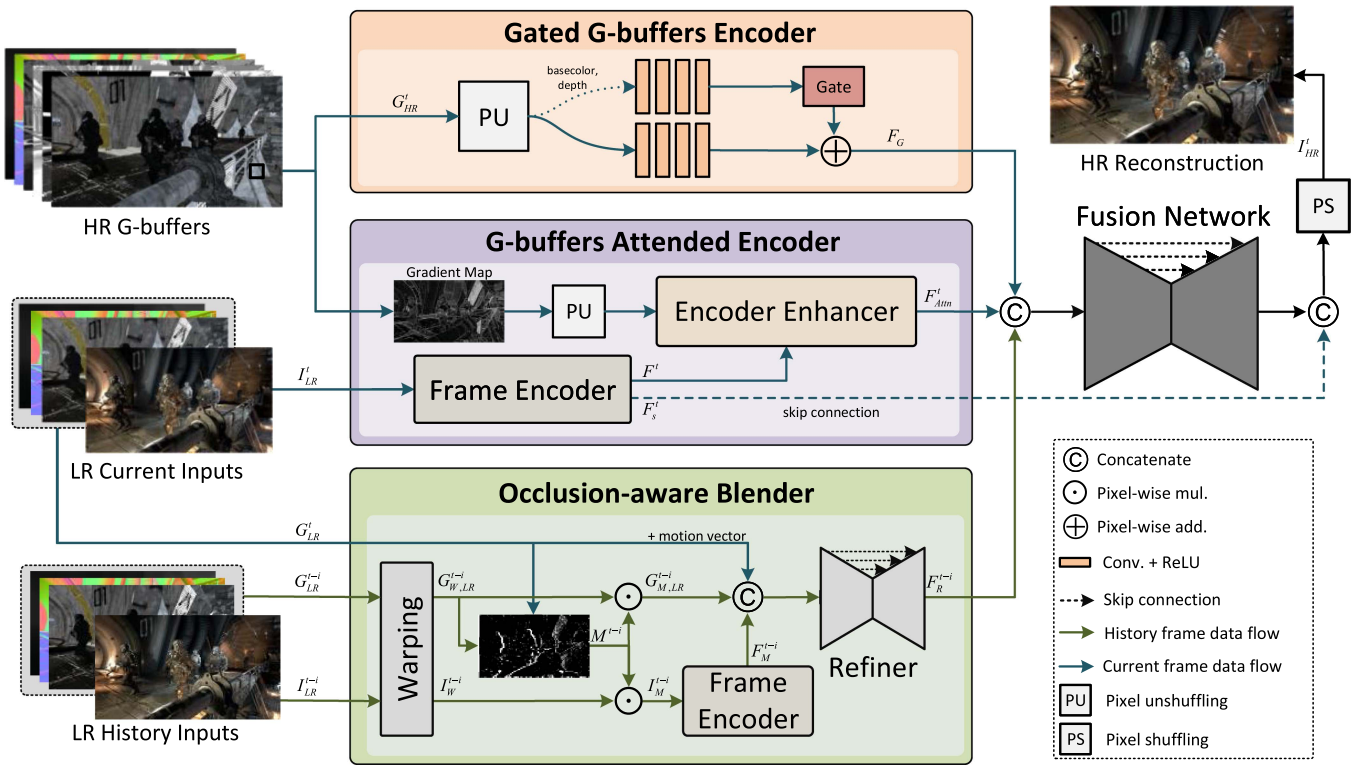


Fig. 2. The pipeline of the proposed method. We first adopt HR G-buffers as auxiliary information, downscale their features to the low-resolution space via a *gated G-buffers encoder*. *G-buffers attended encoder* then enhances the high-frequency information of the current frame features using the HR G-buffers gradient map. For each history frame, an *occlusion-aware blender* computes an occlusion mask based on depth and color comparison, uses the mask to rectify the warped history frame and then refines the history frame features. For visual clarity, we only show one history frame here, but use two history frames in our implementation. Finally, the resulting features of the current and history frames are fused with the G-buffer features via a *fusion network*, and the fused features are passed through an upsampling operation to produce the high-resolution reconstruction.

III. The Method

Although previous works have explored the importance of G-buffers, the information contained in G-buffers has not been fully utilized, making it challenging to achieve high-quality results at larger upsampling ratios (e.g., 4×4). In this paper, our primary goal is to fully extract and leverage detailed information from HR G-buffers to enhance and refine low-resolution image features, thereby significantly improving reconstruction quality by generating realistic high-frequency scene details.

Fig. 2 shows the pipeline of our proposed method. The inputs to our method include: 1) the color image I^t_{LR} and G-buffers G^t_{LR} of the current low-resolution frame; 2) rendered high-resolution G-buffers G^t_{HR} of the current frame, including base-color, metallic, roughness, specular, depth and normal maps; 3) the color image $\{I^{t-i}_{LR}\}^L_{i=1}$ and G-buffers $\{G^{t-i}_{LR}\}^L_{i=1}$ of the history low-resolution frames. Given all this information, our goal is to reconstruct the high-resolution color image I^t_{HR} of I^t_{LR}.

Our model is composed of several *novel components* that are especially tailored for sufficiently harnessing the HR G-buffers information and improving temporal fidelity across high-resolution reconstructions. First, a *gated G-buffers encoder* extracts and downsamples HR G-buffers features to low-resolution, aiming to maximize high-frequency information retention through a gating mechanism that treats different types of G-buffer data unequally. This design differs from FuseSR, which directly fuses high-resolution G-buffer features with low-resolution features after pixel unshuffle. In contrast, our *gated G-buffers encoder* unequally encodes the high-resolution G-buffer features before fusion. Then, a *G-buffers attended encoder* extracts convolutional features of the current frame, and then enhances them with HR gradient map by a cross attention network. Similar to existing supersampling works [4], [5], [10], temporal re-projection is applied to the history frames, warping them to align with the current frame using rendered motion vectors. To address the artifacts in the warped frames due to disocclusion between frames, an *occlusion-aware blender* computes a mask based on the differences between the current depth and color and the history depth and color, which indicates regions that are visible in the current frame but occluded in the previous frame, and applies the mask to produce robust history frame features. Finally, the G-buffer features, the enhanced current frame features and the robust history frame features are sent into a *fusion network* to produce the desired high-resolution reconstruction I^t_{HR}. In the remainder of this section, we describe the key components of our method in detail.

A. Gated G-Buffers Encoder

Considering that G-buffers carry detailed scene information and are relatively cheap to compute, we utilize HR

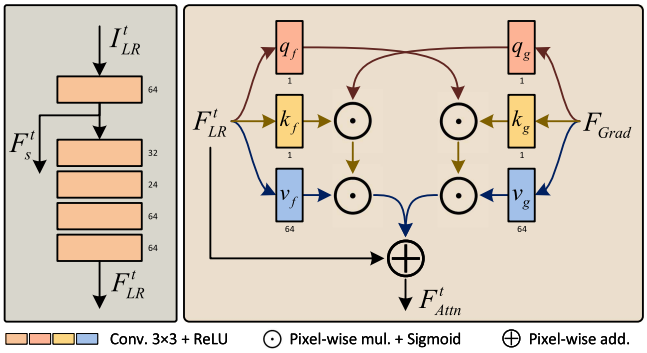


Fig. 3. Left: structure of the frame encoder, which outputs features F^t_{LR} as well as shallow features F^t_s. Right: structure of the encoder enhancer, which leverages cross-attention to fuse the gradient map with the low-resolution features.

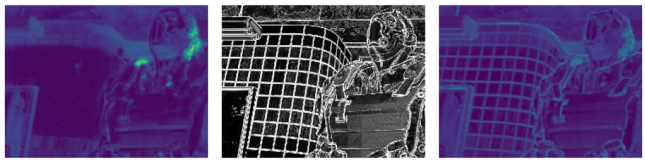
G-buffers as auxiliary information to improve the quality of the high-resolution reconstruction. However, different maps of the G-buffers contribute unequally to the enhancement of image quality. Empirically, the base-color map and depth map are considered more important as they provide features from the perspectives of color and depth values, respectively. This observation is also underpinned by the literature [31].

Based on this, we propose a novel structure—*gated G-buffers encoder*. We first apply pixel unshuffling to downsample the input HR G-buffers by transforming their pixels from the spatial dimension to the channel dimension. Pixel unshuffling can preserve spatial detail information such as textures and edges, as shown in [10]. Subsequently, we propose a dual-branch structure, where each branch consists of multiple conv-relu layers. The first branch encodes the information from base-color and depth maps. The output of this branch is added pixel-wise with the features encoded by the second branch, which utilizes all HR G-buffers to extract complete feature information, as shown in Fig. 2. The first branch processes the basecolor and depth information, which are important features in the G-buffer and thus encoded with a separate branch. However, directly merging these strong features with the output of the second branch may result in them overwhelming other G-buffer features. To address this, a gate is designed to adaptively adjust the fusion of the information from two branches. The gate involves a learnable scaling parameter that is multiplied with the output features of the first branch to control how much information comes from the first branch features in the fusion. The final fused features are represented as F_G.

B. G-Buffers Attended Encoder

We construct a frame encoder consisting of N conv-relu layers to extract convolutional features from the current frame. The frame encoder takes the current frame I^t_{LR} as input and outputs features F^t as well as shallow feature F^t_s for skip connection after the *fusion network*, as shown in Fig. 3. In our experiments, we set N to 4, and F^t_s to be the output of the first conv-relu layer.

LR images often suffer from aliasing issues, which suppress high-frequency signals in the features extracted by the frame encoder, as shown in Fig. 4(a). Hence, it is essential to supplement the low-resolution features with high-frequency features from the HR G-buffer. To effectively enhance the features, rather than directly using G-buffer, we employ a gradient map, which is derived from neighbor pixel difference of base-color and normal maps, as shown in the Fig. 4(b). The gradient map describes the rate of color or luminance changes of an image, where larger gray changes at image edges result in larger gradient magnitudes. We then propose an encoder enhancer that leverages an efficient variant of cross-attention to fuse the gradient map with the low-resolution features, as shown in Fig. 3. Specifically, let F^t_{LR} be the feature map output from the frame encoder and F_{Grad} be the gradient map from the base-color and normal maps of HR G-buffers. In the cross-attention, we adopt a bi-directional attention scheme where F^t_{LR} and F_{Grad} can cross-attend to each other. In particular, we first use F^t_{LR} to calculate the query (q_f) and F_{Grad} for the key (k_g) and value (v_g). Then, F_{Grad} acts as the query (q_g), with F^t_{LR} as the key (k_f) and value (v_f). To reduce computational costs, we use element-wise multiplication as a replacement of matrix multiplication in the standard cross-attention, so that our cross-attention complexity is just linear with respect to the input feature map resolution. The bi-directional cross-attention outputs are finally summed element-wise and added to F^t_{LR} to produce the enhanced map F^t_{Attn} as shown in the Fig. 4(c).



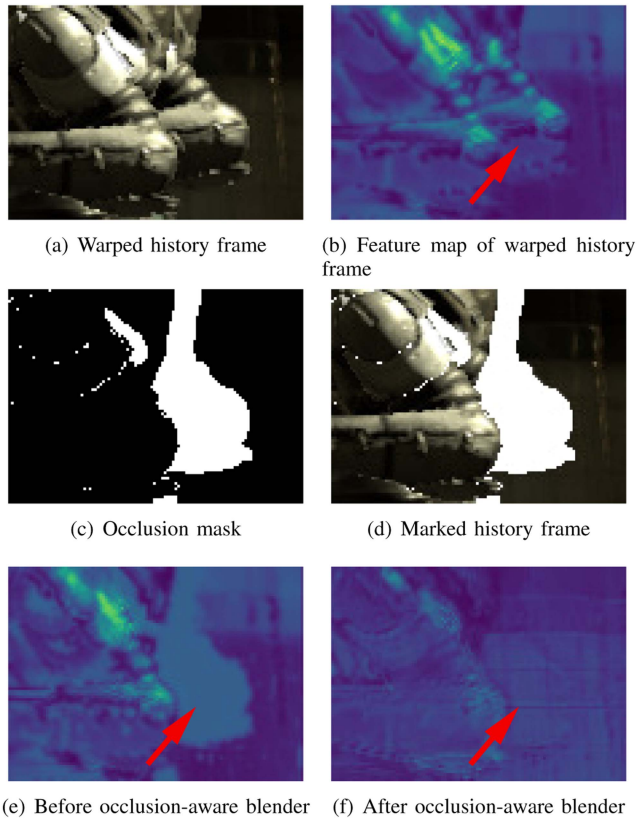
(a) (b) (c)

Fig. 4. The features extracted by the frame encoder (a) lack high-frequency information, and the gradient map (b) computed by using G-buffers can be used to augment the frame encoder output with high-frequency features (c).

C. Occlusion-Aware Blender

Since motion vectors do not consider dynamic dis-occlusion, the temporal re-projection might result in the warped previous frame containing invalid values at regions that are visible in the current frame but occluded in the previous frame due to object motion or viewpoint change, as shown in the Fig. 5(a) and (b). Direct combination of such warped frames with the current frame features will mislead the subsequent reconstruction. To mitigate this issue, previous work [4], [5] trains a small network to predict a pixel-wise weighting map that is used to re-weight all the warped features. This approach has the downside of introducing extra model parameters (thus increasing computational costs) and altering features that are valid and thus expected to be kept unchanged.

To address this problem, we propose a simple and effective *occlusion-aware blender*. This module first obtains a binary occlusion mask M^{t-i} for each warped history frame (Fig. 5(c)), indicating if a position in the warped history frame is dis-occluded or not based on simple depth and color comparisons. Then,



(a) Warped history frame (b) Feature map of warped history frame

(c) Occlusion mask (d) Marked history frame

(e) Before occlusion-aware blender (f) After occlusion-aware blender

Fig. 5. Example results of the occlusion-aware blender. The warped history frame (a) has artifacts at dis-occlusion regions, and direct feature extraction will lead to invalid regions (b). The generated mask (c) based on depth and color comparison identifies invalid positions (white pixels), and is used to rectify the warped history frame to obtain the marked history frame (d). Occlusion-aware blender is able to repair features (e and f) in the marked region using auxiliary G-buffers.

the dis-occluded regions will be marked as invalid (Fig. 5(d)). The frame encoder will extract features only for valid regions (Fig. 5(e)), and invalid regions will be repaired with the help of G-buffers and motion vectors to obtain complete features (Fig. 5(f)).

Specifically, for each warped history frame I_W^{t-i}, we obtain the binary occlusion mask M_D^{t-i} by computing the difference between the depth map D_{LR}^t and the warped depth map D_W^{t-i}. A position x is thought of as being dis-occluded if its difference value is greater than a threshold $\eta_D(x)$:

$$M_D^{t-i}(x) = \begin{cases} 1, & |D_{LR}^t(x) - D_W^{t-i}(x)| > \eta_D(x) \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The 3D points that are closer to the camera, when dis-occluded, will induce smaller depth variations than distant points. Hence, we make $\eta_D(x)$ to be spatially varying by setting $\eta_D(x) = \alpha \cdot D_{LR}^t(x)$.

Although depth comparison can mark most of the invalid regions caused by dynamic dis-occlusion, it can not help identify some invalid regions due to lighting and shadow change between frames. Inspired by the fact that these regions typically exhibit large color difference from their counterparts in the current frame, we additionally introduce binary masks M_B^{t-i} and M_I^{t-i} based on the base-color map and rendered images. For a position x in the warped base-color B_W^{t-i}, we consider a $k_b \times k_b$ local neighborhood around x in B^t. x is marked as invalid if more than 50 % of the pixels in the neighborhood have color distances of larger than a threshold η_B to x. Similarly, we judge the validity of each position in the warped rendered image I_W^{t-i} in the same way, but consider a local neighborhood of size $k_i \times k_i$. Then, we define our final mask M^{t-i} as:

$$M^{t-i} = M_D^{t-i} \mid M_B^{t-i} \mid M_I^{t-i}, \quad (2)$$

where $\mid$ denotes element-wise logical OR. M^{t-i} differentiates between invalid values (1) and valid values (0) in the warped previous frame I_W^{t-i}.

After that, we use M^{t-i} to rectify I_W^{t-i} and the warped G-buffers $G_{W,LR}^{t-i}$ as:

$$I_M^{t-i} = (1 - M^{t-i}) \odot I_W^{t-i}, \quad (3)$$

$$G_{M,LR}^{t-i} = (1 - M^{t-i}) \odot G_{W,LR}^{t-i}, \quad (4)$$

where $\odot$ denotes element-wise multiplication. Afterwards, we employ the frame encoder introduced in Section III-B to extract the features F_M^{t-i} of the warped history image I_M^{t-i}. While masking invalid regions can avoid extracting wrong feature information, it may hurt valid features due to errors in the estimated mask. Therefore, we design a lightweight UNet-based inpainting network, taking as input F_M^{t-i} along with motion vectors, G_{LR}^t and $G_{M,LR}^{t-i}$ as auxiliary information to refine the features of the history frames, and finally obtain the refinement feature F_R^{t-i}.

D. Fusion and Reconstruction

To generate the high-resolution reconstruction, all the features, including F_{Attn}^t, F_G, and $\{F_R^{t-i}\}_{i=1}^L$, are concatenated and fed into the *fusion network* for feature integration. The *fusion network* follows a UNet [11] architecture with skip connections, containing 4 encoder blocks and 4 decoder blocks. The output of the *fusion network* concatenated with the shallow features F_s^t, taken from the frame encoder of the *G-buffers attended encoder*, to obtain the resulting features F_u^t. F_u^t will be upsampled with the pixel shuffling introduced in Section III-A to match the target resolution, and then processed by a single conv-relu layer. This gives rise to the high-resolution reconstruction I_{HR}^t of the current frame.

E. Loss Function

We find that different regions in the reconstruction image may face different challenges. For example, when the normal angle between the current and history frames is too large, the region is likely to suffer from temporal flickering. Also, regions with high metallic or low roughness may lose detailed texture due to excessive illumination.

We update the traditional pixel-wise $L1$ loss and pixel-wise SSIM loss to reflection-aware losses by introducing a binary mask, M_L, to indicate reflective regions. This enables the losses to identify and focus more precisely on these regions during reconstruction. The following regions will be identified by the

binary mask M_L: 1) the region where the angle between the current HR normal and the upsampled history normal is greater than a threshold; 2) the region where the current HR metallic is greater than a threshold; 3) the region where the current HR roughness is less than a threshold. In the loss function, we provide different weighting for the regions based on M_L to pay more attention to the reconstruction quality of these regions.

To train our network, we use the following three loss functions:

- *Pixel-wise weighted L1 loss:* The weighted $L1$ distance is computed between the anti-aliased high-resolution reference image I_{GT} and the reconstructed high-resolution image I_{HR} :

$$\mathcal{L}_{L1} = \|w_{L1} \odot (I_{GT} - I_{HR})\|_1 \tag{5}$$

  where w_{L_1} is a pixel-wise weight map, which is computed as $w_{L_1}(x) = \frac{1}{\sum_x (1+0.5 \cdot M_L(x))}(1 + 0.5 \cdot M_L(x))$ for each position x.
- *Pixel-wise weighted SSIM loss:* We utilize the structural similarity index (SSIM) [32], employing a window size of 11×11:

$$\begin{cases} \mathcal{L}_{SSIM} = \sum_{x \in P_x} S(x) \\ S = w_{SSIM} \odot (\mathbf{1} - SSIM(I_{GT}, I_{HR})) \end{cases} \tag{6}$$

  where w_{SSIM} is a pixel-wise weight map, which is computed as $w_{SSIM}(x) = \frac{1}{\sum_x (1+0.75 \cdot M_L(x))}(1 + 0.75 \cdot M_L(x))$ for position x, and $\mathbf{1}$ is an all-ones map. $SSIM(I_{GT}, I_{HR})$ is also a pixel-wise map, where for each position x, it represents the structural similarity index between I_{GT} and I_{HR} within the window centered at x.
- *LPIPS loss:* We calculate the LPIPS [33] loss between the ground truth and reconstruction:

$$\mathcal{L}_{LPIPS} = LPIPS(I_{GT}, I_{HR}) \tag{7}$$

Our final loss is the weighted sum of the above three losses:

$$\mathcal{L} = \lambda_{L1} \cdot \mathcal{L}_{L1} + \lambda_{SSIM} \cdot \mathcal{L}_{SSIM} + \lambda_{LPIPS} \cdot \mathcal{L}_{LPIPS}, \tag{8}$$

where $\lambda_{L1}, \lambda_{SSIM}$ and λ_{LPIPS} are the weighting parameters to balance different loss terms. We set $\lambda_{L1} = 1.0$, $\lambda_{SSIM} = 1.0$, $\lambda_{LPIPS} = 0.2$. For the M_L, the threshold for the normal angle is set to 45 degrees, the threshold for metallic is set to 0.95, and the threshold for roughness is set to 0.02.

IV. Experiments

A. Dataset

To evaluate our method, we build a dataset using four distinct 3D scenes including *Infiltrator* (*IF*), *Subway* (*SW*), *Redwood Forest* (*RF*) and *Bunker* (*BK*) from Unreal Engine 4 [34]. These scenes feature varying levels of complexity in terms of geometry, material, lighting and motion. *Subway* and *Infiltrator* contain video sequences built by the Unreal official team, covering complex camera movements, character actions, and lighting. The *Bunker* contains more metallic materials and grid-like geometries. The *Redwood Forest* scene consists mainly of leaves and trees with rich high-frequency details. Due to variations in scene size, we render video sequences of different lengths for each scene. In addition, we build a scene *Slum* (*SU*) to verify the generalization ability of the model. Table I presents the lengths of training and test video sequences for each scene, where the training and test sequences do not overlap.

TABLE I
Lengths (Number of Frames) of Training and Test Sequences for Each Scene

Scene	Training	Test
SW	900	120
IF	3700	300
BK	2800	240
RF	3200	240
SU	1200	300

We choose 1920×1080 as the target resolution, and the input resolution is varied according to supersampling ratios. For reference images, we render each image at resolution 1920×1080 with $16\times$ MSAA. For HR G-buffers and low-resolution input images, we do not apply any anti-aliasing. In addition, we enable ray tracing for more realistic visual quality, and disable some post-processing effects such as motion blur, which may cause visual artifacts.

B. Implementation Details and Metrics

We implement our model using PyTorch [35]. We randomly crop the low-resolution inputs into 128×128 patches during training, and run the network on full-resolution input images during test. We utilize the Adam optimizer [36] with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate is configured to $1e-4$, and the batch size is set to 8. We train our model for 200 epoches. For (2), we set $\alpha = 0.1$ when calculating M_D^{t-i}, and set $k_b = 3$, $k_i = 5$ and $\eta_B = 0.5$ when calculating M_B^{t-i} and M_I^{t-i}. The number of the history frames is 2. All the experiments are performed on a NVIDIA GeForce RTX 3090 GPU.

Following [4], [10], we employ peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) [32] as metrics, which are widely used for the evaluation of image reconstruction quality. For PSNR and SSIM, higher values mean better visual quality. Additionally, we evaluate the results using spatio-temporal entropic difference (STRRED) [37], a metric commonly used for video quality assessment that considers temporal stability; lower values are better for STRRED.

The temporal blending method presented in the final part of our video involves the following steps: At frame *t*, the model first obtains the super-resolution result. At frame *t*+1, the same super-resolution result is obtained for the frame *t*+1. Then, using the upsampled motion vector, the super-resolution result of the frame *t* is warped to align with the frame *t*+1. Subsequently, a mask is computed to indicate marked regions, using the same approach as in the occlusion-aware blender. For the marked region, the super-resolution result of frame *t*+1 is computed as a weighted blend: 0.8 · warped frame *t* + 0.2 · frame *t*+1; for the unmarked area, we directly use the super-resolution result of the frame *t*+1 without modification.

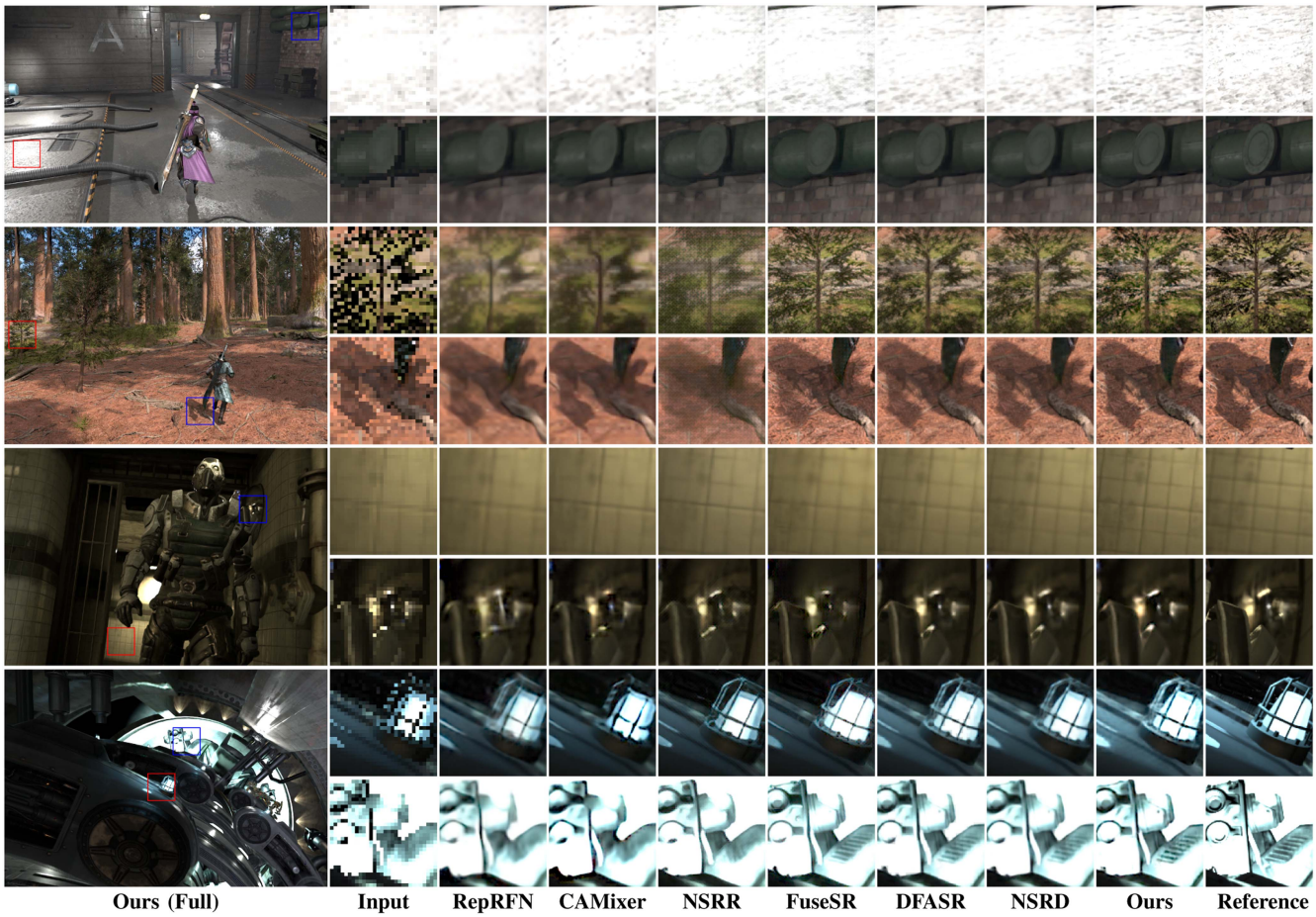


Fig. 6. Visual comparison of different methods for 4×4 upsampling. The results on the *BK*, *RF*, *SW*, and *IF* scenes are shown from top to bottom. Due to a lack of auxiliary information, RepRFN and CAMixer are unable to recover texture details. NSRR leverages LR G-buffer information, performing better on *SW* and *IF* scenes, but its result on the *RF* scene remains unsatisfying. FuseSR and NSRD achieve higher-quality texture reconstruction with the help of HR G-buffers, but its recovery of strong reflections (1st row of *BK*) and shadows (2nd row of *RF*) is not accurate. DFASR suffers from aliasing in the tree (1st row of *RF*) and the floor (1st row of *BK*). In contrast, our reconstructions are realistic, almost on par with the reference images.

C. Quality Evaluation

Comparisons with prior supersampling and super-resolution methods. We compare the proposed method with several existing neural supersampling methods for rendered content, NSRR [4], FuseSR [10], DFASR [16] and NSRD [17], and some image super-resolution works, RepRFN [22] and CAMixer [23]. NSRR utilizes historical frames and LR G-buffers, and FuseSR, DFASR and NSRD utilizes historical frames and HR G-buffers auxiliary information.

Fig. 6 provides a visual comparison of different methods for 4×4 upsampling. NSRR only utilizes simple low-resolution auxiliary information, while RepRFN and CAMixer do not utilize any auxiliary information, making them struggle with reconstructing complex textural details well. In more complex scenes, the abundant geometries lead to significant aliasing, and the lack of guiding auxiliary information for RepRFN and CAMixer from LR G-buffers can lead to low reconstruction quality. Even though NSRR employs LR G-buffers, it still fails to reconstruct the fine-grained texture details. For instance, in the first row of *RF* scene of Fig. 6, the reconstruction of the leaves in the second, third, and fourth columns of the first row is poor. FuseSR employs pixel-unshuffle to merge high-resolution G-buffers with frame features, enhancing scene information and outperforming LR G-buffers based methods. However, it still falls short of fully recovering all details. DFASR achieves arbitrary-scale super-resolution with a single model by modeling frequency domain similarity, but still suffers from aliasing in several scenes, potentially due to its limited fusion module, as shown in the tree in the RF scene and the floor in the BK scene. In contrast, NSRD improves temporal consistency through illumination-material decomposition and recurrent warping, yet struggles to accurately reconstruct shadows of dynamic objects as shown in the 2nd row of RF scene. Our method achieves remarkably better visual quality than FuseSR by more faithfully recovering complex, fine-grained details (e.g., the shadow on the ground in the second row of *RF* scene of Fig. 6), highlighting one of its key advantages. The reconstructions by our method are visually comparable to the reference images. Furthermore, we include the reconstructed video sequences in the supplementary videos, from which it can be observed that our method generates temporally stable results.

We conducted further comparisons with the baseline model FuseSR, and the results are shown in the Fig. 7. Benefiting

TABLE II
QUANTITATIVE COMPARISON OF OUR METHOD WITH EXISTING METHODS FOR 4×4 SUPERSAMPLING ON FOUR DIFFERENT SCENES. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD

Method	PSNR (dB) ↑				SSIM ↑				STRRED ↓			
	BK	RF	SW	IF	BK	RF	SW	IF	BK	RF	SW	IF
RepRFN	25.87	21.12	31.52	24.51	0.8274	0.5425	0.9600	0.8861	1.5424	3.7655	1.6564	1.2498
CAMixer	26.07	20.95	31.22	24.27	0.8452	0.5353	0.9644	0.8900	1.1500	4.1562	1.3567	1.1959
NSRR	25.82	19.50	31.55	24.56	0.8445	0.7401	0.9658	0.8946	1.1570	2.6259	1.2892	0.9461
FuseSR	26.85	25.47	31.58	25.37	0.8821	0.8146	0.9660	0.9065	0.9495	1.8001	1.3014	0.8214
DFASR	27.13	24.29	31.64	**25.97**	0.8851	0.7981	0.9723	**0.9166**	0.9830	2.2617	1.2692	0.7401
NSRD	26.71	24.34	32.07	25.02	0.8722	0.8047	**0.9742**	0.9013	0.8768	1.5591	**0.9513**	0.7855
Ours	**27.69**	**25.93**	**32.29**	25.66	**0.9039**	**0.8229**	0.9720	0.9148	**0.6050**	**0.9503**	1.0400	**0.6079**

TABLE III
QUANTITATIVE COMPARISON OF OUR METHOD WITH COMMERCIAL METHODS FOR 2×2 SUPERSAMPLING ON FOUR DIFFERENT SCENES. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD

Method	PSNR (dB) ↑				SSIM ↑				STRRED ↓			
	BK	RF	SW	IF	BK	RF	SW	IF	BK	RF	SW	IF
FSR	26.58	20.46	32.46	26.83	0.8983	0.6248	0.9731	0.9427	0.6052	2.1217	**0.7934**	0.2572
DLSS	27.98	21.01	32.04	26.77	0.9322	0.6769	**0.9814**	**0.9551**	0.4567	2.0674	0.8909	0.3271
XeSS	27.05	20.69	31.39	26.95	0.9002	0.6253	0.9772	0.9442	0.4785	2.2778	0.8588	0.2609
TAAU	25.36	20.64	31.38	26.52	0.8982	0.6221	0.9772	0.9421	0.5131	2.3278	0.8604	0.2867
Ours-2×	**29.52**	**27.18**	**32.74**	**28.15**	**0.9418**	**0.8627**	0.9735	0.9470	**0.3171**	**0.5382**	0.9287	**0.2389**

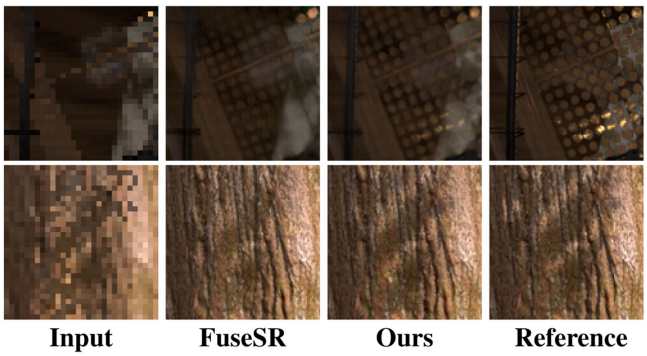


Fig. 7. Extended visual comparisons with FuseSR for 4×4 upsampling. Our method achieves better edge and shadow details. The results on the *IF* and *RF* scenes are shown in the first row and second row, respectively.

from the *G-buffers attended encoder's* effective enhancement of image feature edges, our method achieves sharper edge details in the reconstruction of the first-row of the Fig. 7. Meanwhile, the *occlusion-aware blender* can fully and accurately leverage temporal information to correct color correspondences. Even without geometric buffer guidance, our method delivers higher visual quality in shadow reconstruction, as illustrated in the second-row images of the Fig. 7.

The quantitative results are shown in Table II. Our method achieved the best PSNR and SSIM scores across all the scenes, demonstrating its ability to produce superior image reconstruction quality. Additionally, the STRRED scores show that our method also achieves superior video quality compared to the other methods, thanks to our carefully designed *occlusion-aware blender*, which provides more accurate historical features.

Comparison with commercial techniques. In the gaming and movie industries, many popular supersampling techniques for real-time rendering are available, including NVIDIA DLSS 2 [1], AMD FSR 2 [2], Intel XeSS [3], and Epic TAAU [15]. Although some of these methods can achieve supersampling at 3×3 or higher upscaling ratios, the reconstructed images are often too blurry, limiting their broader adoption. Considering the balance between reconstruction quality and performance, we compare our method with these methods for 2×2 supersampling, where we use official plugins in Unreal Engine to sample the upsampled frame sequences by these methods from all the scenes.

The quantitative results are presented in Table III. Leveraging the rich scene information in high-resolution G-buffers, our method faithfully reconstructs texture details, achieving the highest PSNR values and excellent SSIM scores across all scenes. Additionally, the STRRED scores demonstrate that our method provides superior temporal stability at 2×2 resolution, thanks to our occlusion-aware blender, which efficiently corrects historical frames to obtain valid temporal information. Specifically, the *RF* dataset contains a large amount of dynamic foliage, and the *BK* dataset features numerous complex meshes with metallic materials, both of which cause significant aliasing in the low-resolution images.

Fig. 8 provides a visual comparison of different methods. Compared with FSR, which does not utilize neural networks, DLSS delivers higher quality in reconstructing scene details and significantly reduces artifacts. Although the performance of TAAU and XeSS is not as strong as that of DLSS and FSR, they still manage to achieve good reconstruction quality on most scenes. However, in complex scenes where complex geometries are present, TAAU produces severe undersampling issues. Even with the use of historical frame information, the high-resolution images reconstructed by these methods still suffer from significant blurriness or texture errors. For example, in Fig. 8, the first row of *BK* scene shows that the existing methods exhibit noticeable aliasing, while their results appear relatively blurry in

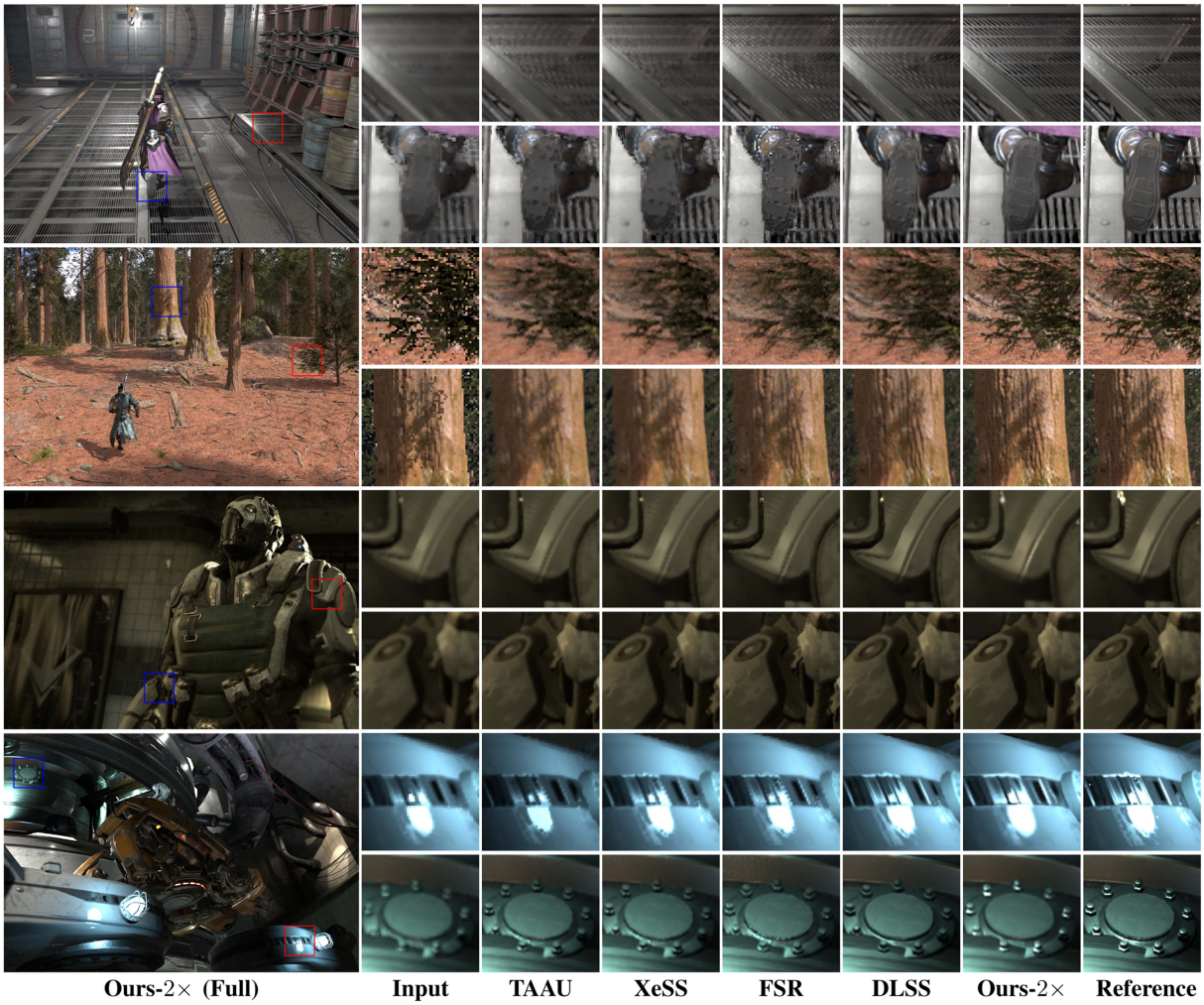


Fig. 8. Visual comparison of different methods for 2×2 upsampling. The results for the *BK*, *RF*, *SW*, and *IF* scenes are shown from top to bottom. The other approaches exhibit noticeable aliasing issues on the *BK* and produce blurred reconstructions on the *RF*. In contrast, our method can produce highly realistic and sharp results. On the *IF*, the dramatic changes in lighting lead to the incorrect light reconstructions of the other methods (1st row of the *IF*), whereas our method is robust to the lighting change due to the occlusion-aware blender and correctly predicts the brightness of the light. Furthermore, due to our extra focus on strong reflection areas, our method achieves significantly better performance in highlight reconstruction (2nd row of the *IF*).

the first row of *RF*. HR G-buffers retain more scene information due to their higher sampling rate. Our designed *Gated G-buffers encoder* enhances the image reconstruction by providing rich texture details, while the *G-buffers attended encoder* effectively preserves image edges, which becomes especially prominent in highly complex scenes. Additionally, our method relies on two history frames and utilizes a lightweight UNet-based refiner network to correct invalid information, making it more effective for rapidly changing scenes. For example, in the first row of *IF* scene, the brightness of the lights changes dramatically between adjacent frames, but our method is still able to achieve accurate reconstruction. As a result, our method achieves the outstanding reconstruction quality in all the scenes.

D. Runtime

Following the previous works [4], [10], we use NVIDIA TensorRT [38] to optimize our trained model in 16-bit floating point mode. In Table IV, we report the runtime of our method, FuseSR and NSRR for 4×4 supersampling on different target resolutions. The runtime is measured in unit of milliseconds (ms). The results show that our method runs significantly faster than NSRR, partially because our method needs less history frames than NSRR. In addition, NSRR upsamples the history frames to the target (high) resolution prior to warping and feature

TABLE IV
RUNTIME (MS) COMPARISON OF OUR METHOD WITH PRIOR SUPERSAMPLING METHODS FOR 4×4 UPSAMPLING ON DIFFERENT TARGET RESOLUTIONS: 1280×720 (720P), 1920×1080 (1080P), 2560×1440 (2K) AND 4096×2160 (4K)

Method	720p	1080p	2K	4K
Ours	3.43	8.01	14.96	35.34
FuseSR	3.28	8.79	15.03	33.97
NSRR	10.64	23.90	43.30	-

TABLE V
RUNTIME (MS) OF EACH COMPONENT OF OUR MODEL FOR 4×4 SUPERSAMPLING ON 1920×1080 TARGET RESOLUTION. THE HR G-BUFFERS RENDERING TIME IS OBTAINED BY AVERAGING ACROSS ALL THE SCENES

Module	Time (ms)
HR G-buffers render	0.97
HR G-buffers encoder	2.45
G-buffers attended encoder	0.53
Occlusion-aware blender	1.81
Fusion network	3.06
Reconstruction	0.24

TABLE VI
ABLATION STUDY RESULTS ON HR G-BUFFER CHANNELS. ALL THE SCORES ARE MEASURED ON THE *IF* SCENE FOR 4×4 UPSAMPLING

Method	PSNR (dB)	SSIM
w/o basecolor	25.02	0.9090
w/o normal	25.06	0.9100
w/o metallic	25.24	0.9112
w/o roughness	25.26	0.9114
w/o specular	25.34	0.9122
w/o depth	25.32	0.9117
w depth	24.59	0.9004
Ours	25.66	0.9148

TABLE VII
QUANTITATIVE COMPARISON BETWEEN OUR MODEL AND FUSESR WHEN TRAINED ON EACH SCENE SEPARATELY (P) AND ON ALL THE SCENES JOINTLY (J), RESPECTIVELY. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD

		Ours-P	Ours-J	FuseSR-P	FuseSR-J
PSNR (dB)	BK	**27.69**	27.44	26.85	26.91
	RF	**25.93**	25.41	25.47	25.44
	SW	**32.29**	32.28	31.58	31.28
	IF	**25.66**	24.56	25.37	25.27
SSIM	BK	**0.9039**	0.8967	0.8821	0.8838
	RF	**0.8229**	0.8074	0.8146	0.8042
	SW	**0.9720**	0.9716	0.9660	0.9616
	IF	**0.9148**	0.8946	0.9065	0.9055

TABLE VIII
QUANTITATIVE COMPARISON OF OUR MODEL AND FUSESR WHEN TRAINED ON *SU* SCENE (SU) AND ON ALL THE SCENES JOINTLY (J), RESPECTIVELY, AND EVALUATED ON AN UNSEEN SCENE. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD

	Ours-SU	Ours-J	FuseSR-SU	FuseSR-J
PSNR (dB)	**23.77**	22.90	23.47	22.16
SSIM	**0.8591**	0.8395	0.8501	0.8081

rectification, causing higher computational costs that our method that processes the history frames at the input (low) resolution. FuseSR utilizes simple pixel operations to implement downsampling of HR G-buffers, thus achieving higher inference speed. In contrast, our method learns a much more sophisticated *gated G-buffers encoder*, which allows us to better employ information from HR G-buffers for significantly improved visual quality, at the cost of slightly increased runtime. However, we employ a UNet-based fusion network, where all the blocks operate on features of spatial resolutions lower than the input resolution except for the first and last blocks, as opposed to FuseSR's fusion network that processes features of the input resolution in every block. This significantly reduces the overall inference time. As a result, our method is almost as fast as or faster than FuseSR, while achieving dramatically improved visual and temporal fidelity, as shown in Table II. We further report the runtime of each component in our model in Table V.

E. HR G-Buffer Channels

We ablate different channels in the HR G-buffer to investigate their impact on the performance of our model. This study aims to provide insight into what HR G-buffer channels are important for our model, as well as empirical evidence for some important design choices in our model. We also consider a simple variant that only uses depth information as auxiliary information, which can greatly reduce memory consumption and thus makes our method applicable to forward rendering.

Table VI shows that different HR G-buffer channels have varying impacts on the reconstruction quality. Removing the basecolor results in the most significant quality degradation. This is because the basecolor stores the scene color information without lighting, making it similar to the rendered image and thus providing accurate texture information. For this reason, when designing the *G-buffers encoder*, we specifically create a separate branch to extract color features independently as the primary features. The normal is used for extracting edge features, strengthening the edge information in low-resolution frame features. Additionally, the normal provides information about highly reflective areas for the loss calculation. Thus, removing the normal leads to a noticeable decline in reconstruction quality. In contrast, the roughness, metallic, and specular are single-channel G-buffers that carry less scene information; their absence has relatively little impact. When only the depth map is provided as auxiliary information, the model performance decreases noticeably. This is due to the significant reduction in available color and reflection information, the unavailability of reflection-aware loss during training, which limits the model's ability to refine local scene details.

F. Generalizability

All the experiments so far are based on training our model separately for each scene and testing it on the same scene, following a common practice in the existing works [4], [10]. We are interested in how well our model can generalize across different scenes. To investigate this, we experiment with training our network on the training data of *all* the scenes, and test it on: 1) the test data of all the training scenes and 2) the data of an unseen *Slum* (*SU*) scene. The *SU* is an open market scene consisting of numerous buildings with complex textures.

Tables VII and VIII report the results under the two test settings, respectively, where we observe a similar trend as follows. For our model, training on all the scenes performs slightly worse than per-scene training, as also observed in existing works [4], [10]. Our model trained on all the scenes significantly outperforms FuseSR trained under the same setting, and even compares favorably with FuseSR trained for each scene separately. This

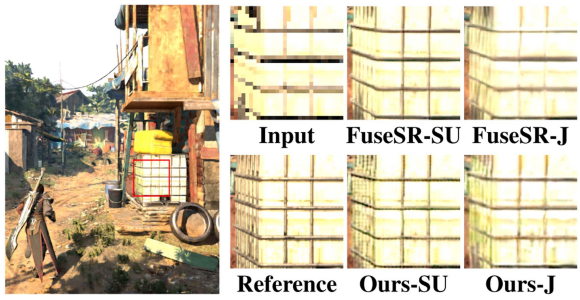


Fig. 9. Visual comparison of our method and FuseSR on an unseen scene. "SU" and "J", respectively, represent training for each scene separately and on all the scenes jointly.

TABLE IX
DESIGN ANALYSIS FOR VARIOUS COMPONENTS OF OUR METHOD. RESULTS IN THE TABLE ARE MEASURED ON THE BUNKER SCENE. THE BEST RESULTS ARE IN BOLD

	PSNR (dB)	SSIM
w/o HR G-buffers	26.40	0.8625
w/o gated G-buffers encoder	27.26	0.8919
w/o gated branch	27.31	0.8930
w/o encoder enhancer	27.07	0.8887
w/ single-direction coss-attention	27.40	0.8944
w/o occlusion-aware blender	27.23	0.8907
w/ learned reweighting	27.24	0.8911
w/o refiner	27.28	0.8918
w/o pixel-wise loss weights	27.32	0.8930
Ours	**27.69**	**0.9039**

suggests that our model is able to generalize across different scenes successfully, with better generalization ability than FuseSR. Fig. 9 provides a visual comparison of our method and FuseSR.

G. Design Analysis

The superior performance of our network is attributed primarily to several important architectural and loss designs that are especially tailored to effectively harness information in HR G-buffers. We conduct ablation experiments to explore different design choices in our model and show the results in Table IX.

HR G-buffers. Using LR G-buffers instead of HR G-buffers inputs (w/o HR G-buffers) leads to significantly reduced performance, confirming the importance of using high-frequency information in HR G-buffers in our model. Although being commonly used auxiliary information in previous works, LR G-buffers suffer from significant aliasing. Images reconstructed using LR G-buffers often exhibit noticeable blurriness and lack detailed textures. In contrast, using HR G-buffers effectively avoids this issue.

Gated G-buffers encoder. We try replacing our *gated G-buffers encoder* with just a simple downsampling operation with pixel unshuffling (w/o gated G-buffers encoder), which is used in FuseSR [10], and find this simple alternative performs worse. Such simple downsampling operation can match the spatial resolution of the HR G-buffers with that of the low-resolution input frames while preserving detail information. However, this method struggles to fully extract features from HR G-buffers, resulting in degraded visual quality. In contrast, we design a *gated G-buffers encoder*, which effectively removes redundant information across various G-buffer *channels* and uses a dual-branch structure to more efficiently extract features and enhance color details, thereby achieving better recovery of texture details. Additionally, our *gated G-buffers encoder* includes a gated branch where important base-color and depth information can be selected adaptively through a gating mechanism before being merged with the output of the main branch. We find that removing this gated branch (w/o extra branch) leads to a significant decline in reconstruction quality.

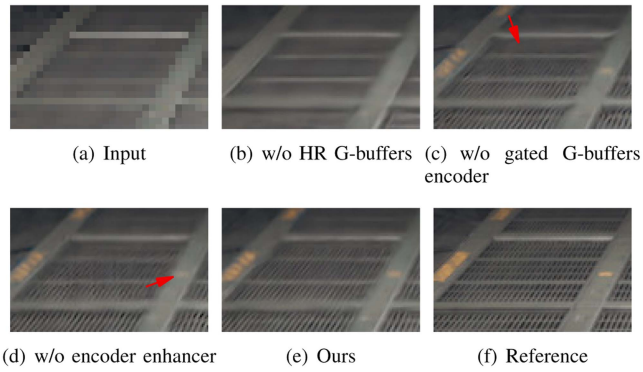
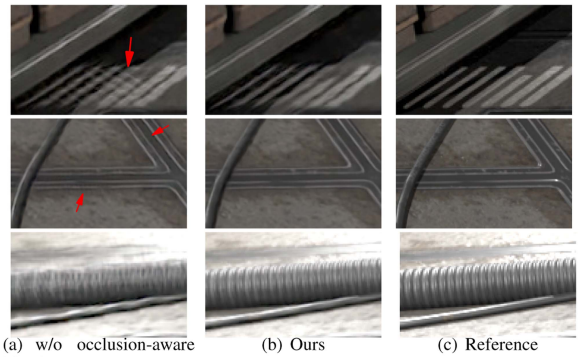


Fig. 10. Ablation study on HR G-buffers, the *gated G-buffers encoder* and the *G-buffers attended encoder*. Without HR G-buffers, the high-frequency texture details are lost in the reconstruction. When the *gated G-buffers encoder* is disabled or the gradient map of HR G-buffers in the *G-buffers attended encoder* is not utilized to enhance the LR current frame features, the recovered texture has obvious distortion.

G-buffers attended encoder. Removing the HR G-buffers gradient and cross attention module (w/o encoder enhancer), the current frame features will not be augmented by the high-frequency information from HR G-buffers, resulting in the loss of texture details in the reconstruction and thus degraded quality. Fig. 10 compares the results with and without the encoder enhancer visually. We also explore replacing the bi-direction cross-attention in the encoder enhancer with a simple single-direction variant (w/ single-direction cross-attention) to fuse F_{LR}^{t} and F_{Grad}, where F_{LR}^{t} acts as the query and F_{Grad} acts as both the key and value. The experimental result indicates that this leads to a slight degradation in reconstruction quality, suggesting that our proposed lightweight cross-attention is effective.

Occlusion-aware blender. Direct use of the warped history frames to extract features without considering their invalid values due to dynamic dis-occlusion or shading changes refining their features (w/o occlusion-aware blender) produces worse results, as shown in the Fig. 11, suggesting the necessity of our occlusion-aware blender. To test the effect of our occlusion mask based on simple depth and color comparison, we consider using a learned reweighing module, i.e., the attention in FuseSR, to compute the weights for the history frames (w/ learned reweighing), and find that it reduces reconstruction quality. The lightweight UNet-based refiner can leverage G-buffers and motion vectors to repair missing information in the history frame features.



(a) w/o occlusion-aware blender (b) Ours (c) Reference

Fig. 11. Ablation study on the *occlusion-aware blender*. The model without the *occlusion-aware blender* results in the wrong shadow.

Removing the refiner and replacing it with a simple conv-relu layer (w/o refiner) would reduce the reconstruction quality.

Reflection-aware loss. The extra pixel-wise weights in the loss function help the network focus on challenging regions, such as regions of smooth surfaces and strong reflection. These regions are prone to losing detailed textures and may even suffer from distortions, thus causing a decline in the quality of the reconstructed images if not treated specially. Removing the extra pixel-wise weights in the L_1 and SSIM losses leads to a performance drop.

V. Limitations and Future Work

Despite its promising performance, particularly on recovering rich, high-frequency scene details, our method has three limitations, which leaves room for future research.

First, using HR G-buffer requires storing and processing high-resolution maps, which is expensive in terms of compute and memory. According to Burns et al. [39], the G-buffer for a single visibility sample consumes over 20 bytes. This imposes significant demands on memory bandwidth and geometry processing, making it challenging for low-end GPUs to handle—particularly in highly complex scenes. Although our method is optimized for real-time computational efficiency, it is currently infeasible to be deployed on mobile or integrated low-end GPUs, primarily due to the high demands of high-resolution G-buffer storage and computation. However, further optimization may be explored in future work to address this. Burns et al. designed a visibility buffer which holds a triangle index as well as an instance ID for each sample and accesses the triangle data through this index to compute vertex attributes in deferred shading. This approach avoids the bandwidth and memory costs associated with transferring and storing the G-buffer while enabling more flexible programming of the rendering pipeline. For future work, we can build on their cache-friendly concept and visibility buffer to develop more compact storage structures or optimize their usage in our upsampling process. In addition to this, there are also some industry methods such as DLSS [1], FSR [2], and XeSS [3], which may employ various engineering techniques to improve their real-time performance. In future work, to further increase efficiency, we will focus on systematic research regarding the industrial adaptation of models, with key priorities including lightweight network architecture design, data transmission optimization, and computational efficiency enhancement. Specifically, to address computational bottlenecks on low-end GPU platforms, implementing a frame buffer tiling strategy [40] could effectively alleviate processing constraints. For non-NVIDIA GPU platforms, cross-hardware compatibility will be improved through heterogeneous computing optimization of core modules via customized compute shaders.

(a) Input (b) Ours (c) Reference

Fig. 12. Typical failure case of our method in reconstructing fine texture details at greater distances.

(a) Input (b) Ours (c) Reference

Fig. 13. Typical failure case of our method when reconstructing complex shadow details.

Second, high-resolution G-buffers may fail to provide meaningful features in certain scenarios—for example, in the presence of fog, shadows, refraction, or complex textures on distant objects. In such cases, over-reliance on G-buffer information can lead to inaccurate texture reconstruction. As illustrated in Figs. 12 and 13, we present two failure cases under these challenging conditions. In Fig. 12, our method still exhibits noticeable blurriness on distant objects, despite the use of high-resolution geometric buffers. In Fig. 13, the reconstruction quality degrades in regions with complex shadows, resulting in a lack of fine detail. To address this limitation, a promising direction for future work is to develop a scene-aware network that can adaptively extract and fuse G-buffer features based on contextual cues.

Third, our method sometimes suffers from flickering artifacts. Although our method demonstrates superior STRRED scores compared to the baselines, our results on challenging scenarios, such as the Redwood Forest scene, still exhibit some flickering artifacts, although this issue is also present in all the baseline methods. The flickering issue is less noticeable for commercial methods, possibly because the commercial approaches blend the current frame and historical frames in pixel space (i.e., blend their pixel colors). The pixel-level temporal blending helps reduce flickering, but at the cost of potential blurring. As shown in the supplementary video, despite being temporally coherent, the results of FSR and TAAU lack high-frequency details. In contrast, our method learns to fuse the features of current frame and historical frames, which are then used to reconstruct

the high-resolution image in pixel space. Such learning-based feature-level fusion significantly improves visual fidelity, but may cause slight flickering partially due to the reconstruction error. To address the flickering issue, one possible way is to apply pixel-level temporal blending, on the high-resolution reconstructed image along with reconstructed history frames. At the end of the supplementary video, we present the results of integrating such temporal blending technique into our pipeline, which show improved temporal coherence with slight reduction in visual quality.

Furthermore, in future work, modeling the relationship between rendered images and geometric buffer features by incorporating other feature domains (such as the frequency domain and gradient domain) is a promising direction for exploration. Frequency domain-based methods have already shown potential in previous super-resolution research. For example, Zhou et al. [41] achieved effective upsampling through frequency-domain modeling. Additionally, SGNet [42] utilizes both gradient and frequency domains to perform depth-guided image super-resolution. More recently, DFASR [16] explored the similarity between rendered images and geometric buffers in the frequency domain, establishing feature correlations in this domain and applying them to real-time arbitrary-scale rendering super-resolution tasks. These methods aim to leverage additional information in various image domains to enhance reconstruction quality. However, challenges still remain in terms of real-time performance, feature alignment, and stability in dynamic scenes. In the future, we aim to identify the shared and domain-specific characteristics of rendered images and geometric buffer features across different domains, fusing similar features while enhancing image details using the unique properties of geometric buffers. This will enable more efficient and accurate cross-domain fusion strategies, ultimately improving the effectiveness of high-resolution G-buffers in complex dynamic scenes.

VI. Conclusion

In this paper, we propose a supersampling neural network for rendered content, which has excellent capabilities to fully utilize low-cost high-resolution G-buffers through several well designed fundamental architectural components, including a *gated G-buffers encoder* and a *G-buffers attended encoder*, along with a novel *reflection-aware loss*. Meanwhile, in order to improve temporal coherence, we propose an *occlusion-aware blender*, which employs depth and color information to de-select invalid features in the warped history frames. Experimental results show that our method is particularly effective in reconstructing high-frequency scene details, significantly outperforming the prior supersampling methods while being still compute-efficient. Furthermore, our performance matches or surpasses that of some popular commercial techniques that rely on proprietary models and data.

Acknowledgment

Chengzhi Yuan conducted all experiments for both the original and revised submissions. Zhaocheng You contributed by assisting with GPU data collection and manuscript editing.

References

[1] NVIDIA, "Deep learning super sampling (DLSS) technology | NVIDIA," 2018. [Online]. Available: https://www.nvidia.com/en-us/geforce/technologies/dlss/
[2] AMD, "AMD FidelityFX^{TM} super resolution," 2021. [Online]. Available: https://www.amd.com/en/technologies/fidelityfx-super-resolution/
[3] Intel, "XeSS super sampling," 2022. [Online]. Available: https://www.intel.com/content/www/us/en/products/docs/discrete-gpus/arc/technology/xess.html
[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, 2020, Art. no. 142.
[5] S. Yang et al., "MNSS: Neural supersampling framework for real-time rendering on mobile devices," *IEEE Trans. Vis. Comput. Graphics*, vol. 30, no. 7, pp. 4271–4284, Jul. 2024.
[6] B. Xu et al., "Adversarial Monte Carlo denoising with conditioned auxiliary feature modulation," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 224:1–224:12, 2019.
[7] B. Zhang and H. Yuan, "High-quality real-time rendering using subpixel sampling reconstruction," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 7006–7014.
[8] J. Guo et al., "ExtraNet: Real-time extrapolated rendering for low-latency temporal supersampling," *ACM Trans. Graph.*, vol. 40, no. 6, pp. 278:1–278:16, 2021.
[9] S. Wu et al., "ExtraSS: A framework for joint spatial super sampling and frame extrapolation," in *Proc. ACM SIGGRAPH Asia Conf. Papers*, 2023, pp. 92:1–92:11.
[10] Z. Zhong et al., "FuseSR: Super resolution for real-time rendering through efficient multi-resolution fusion," in *Proc. ACM SIGGRAPH Asia 2023 Conf. Papers*, 2023, pp. 8:1–8:10.
[11] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput. Assist. Interv.*, 2015, pp. 234–241.
[12] K. Akeley, "Reality engine graphics," in *Proc. 20th Annu. Conf. Comput. Graph. Interactive Techn.*, 1993, pp. 109–116.
[13] B. Karis, "High quality temporal anti-aliasing," in *Proc. Adv. Real-Time Rendering Games SIGGRAPH Courses*, 2014.
[14] L. Yang, S. Liu, and M. Salvi, "A survey of temporal antialiasing techniques," *Comput. Graph. Forum*, vol. 39, no. 2, pp. 607–621, 2020.
[15] Epic, "Screen percentage with temporal upscale in unreal engine," 2020. [Online]. Available: https://docs.unrealengine.com/en-US/screen-percentage-with-temporal-upscale-in-unreal-engine/
[16] H. Zhang et al., "Deep Fourier-based arbitrary-scale super-resolution for real-time rendering," in *Proc. ACM SIGGRAPH Conf. Papers*, 2024, Art. no. 65.
[17] J. Li, Z. Chen, X. Wu, L. Wang, B. Wang, and L. Zhang, "Neural super-resolution for real-time rendering with radiance demodulation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 4357–4367.
[18] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 184–199.
[19] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 1132–1140.
[20] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2472–2481.
[21] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 294–310.
[22] W. Deng, H. Yuan, L. Deng, and Z. Lu, "Reparameterized residual feature network for lightweight image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 1712–1721.
[23] Y. Wang, Y. Liu, S. Zhao, J. Li, and L. Zhang, "CAMixerSR: Only details need more "attention"," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 25837–25846.
[24] J. Caballero et al., "Real-time video super-resolution with spatio-temporal networks and motion compensation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2848–2857.
[25] X. Wang, K. C. K. Chan, K. Yu, C. Dong, and C. C. Loy, "EDVR: Video restoration with enhanced deformable convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 1954–1963.

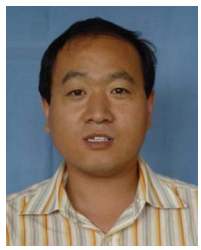
[26] Y. Jo, S. W. Oh, J. Kang, and S. J. Kim, "Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3224–3232.
[27] Y. Chen, S. Chen, Y. Chen, Y. Lin, and W. Peng, "MoTIF: Learning motion trajectories with local implicit neural functions for continuous space-time video super-resolution," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 23074–23084.
[28] M. Liu, S. Jin, C. Yao, C. Lin, and Y. Zhao, "Temporal consistency learning of inter-frames for video super-resolution," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 4, pp. 1507–1520, Apr. 2023.
[29] E. Ioannou and S. Maddock, "Towards real-time G-buffer-guided style transfer in computer games," *IEEE Trans. Games*, vol. 17, no. 3, pp. 613–621, 2025.
[30] X. Wei, H. Huang, Y. Shi, H. Yuan, L. Shen, and J. Wang, "End-to-end adaptive Monte Carlo denoising and super-resolution," 2021, *arXiv:2108.06915*.
[31] F. Rousselle, M. Manzi, and M. Zwicker, "Robust denoising using feature and color information," *Comput. Graph. Forum*, vol. 32, no. 7, pp. 121–130, 2013.
[32] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[33] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 586–595.
[34] Epic, "Unreal engine," 2021. [Online]. Available: Unreal Engine. https://www.unrealengine.com/
[35] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.
[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.
[37] R. Soundararajan and A. C. Bovik, "Video quality assessment by reduced reference spatio-temporal entropic differencing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 4, pp. 684–694, Apr. 2013.
[38] NVIDIA, "NVIDIA TensorRT," 2018. [Online]. Available: https://developer.nvidia.com/tensorrt/
[39] C. A. Burns and W. A. Hunt, "The visibility buffer: A cache-friendly approach to deferred shading," *J. Comput. Graph. Techn.*, vol. 2, no. 2, pp. 55–69, 2013.
[40] L. Seiler et al., "Larrabee: A many-core x86 architecture for visual computing," *IEEE Micro*, vol. 29, no. 1, pp. 10–21, Jan./Feb. 2009.
[41] M. Zhou et al., "Deep Fourier up-sampling," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2022, Art. no. 1671.
[42] Z. Wang, Z. Yan, and J. Yang, "SGNet: Structure guided network via gradient-frequency awareness for depth map super-resolution," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 5823–5831.

Pengjie Wang received the PhD degree in computer science from Zhejiang University. He is currently a professor with the School of Computer Science, Dalian Minzu University, Dalian, China. He is also with the National Center for Computer Animation, Faculty of Media and Communication, Bournemouth University, U.K. His research interests include computer vision and computer graphics.

Chengzhi Yuan is currently working toward the master's degree with the School of Information and Communication Engineering, Dalian Minzu University, Dalian, China. His research interests include deep learning, computer vision, and computer graphics.

Jie Guo received the PhD degree from Nanjing University, in 2013. He is an associate professor (with tenure) with the School of Computer Science, Nanjing University. His current research interest is mainly in computer graphics, virtual reality, and 3D vision. He has more than 100 publications in internationally leading conferences (SIGGRAPH, SIGGRAPH Asia, CVPR, ICCV, ECCV, IEEE VR, etc.) and journals (the *ACM Transactions on Graphics*, *IEEE Transactions on Visualization and Computer Graphics*, *IEEE Transactions on Image Processing*, etc.).

Xiaosong Yang is currently a professor, deputy head of department, programme leader of MSc AIM with the National Center for Computer Animation, Bournemouth University, Bournemouth, U.K. He has more than 30 years' experience of research, education and professional practice in computer animation, machine learning, data mining, digital health, virtual reality, and surgery simulation.



Houjie Li is currently a professor with the School of Information and Communication Engineering, Dalian Minzu University, Dalian, China. His research interests include deep learning, computer vision, and image processing.

Ian Stephenson is a senior lecturer with the National Center for Computer Animation, Bournemouth University where he teaches computing within an art based framework. Through the University's Center for Excellence in Media Practice, and STEM outreach programs he is also lead developer on the Sniff, post-Scratch programming language.

Jian Chang is an active scientist in computer animation with more than 15 years' research experience with the National Center for Computer Animation, Faculty of Media and Communication, Bournemouth University, U.K. His research has focused on physics based modeling, motion synthesis, virtual reality (surgery simulation), and novel HCI (eye tracking, gesture control, and haptic).

Ying Cao received the BEng and MSc degrees in software engineering from Northeastern University, China, and the PhD degree in computer science from the City University of Hong Kong. He is an assistant professor with ShanghaiTech University. His general research interests include computer graphics and computer vision, with a special interest in data-driven graphic design.