Deep Fourier-based Arbitrary-scale Super-resolution for Real-time Rendering

Haonan Zhang*
502022330066@smail.nju.edu.cn
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, China

Jie Guo*
guojie@nju.edu.cn
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, China

Jiawei Zhang
zjw114592814@163.com
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, China

Haoyu Qin
hao19981208@gmail.com
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, China

Zesen Feng
zs_feng@smail.nju.edu.cn
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, China

Ming Yang
502022330060@smail.nju.edu.cn
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, China

Yanwen Guo†
ywguo@nju.edu.cn
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, China

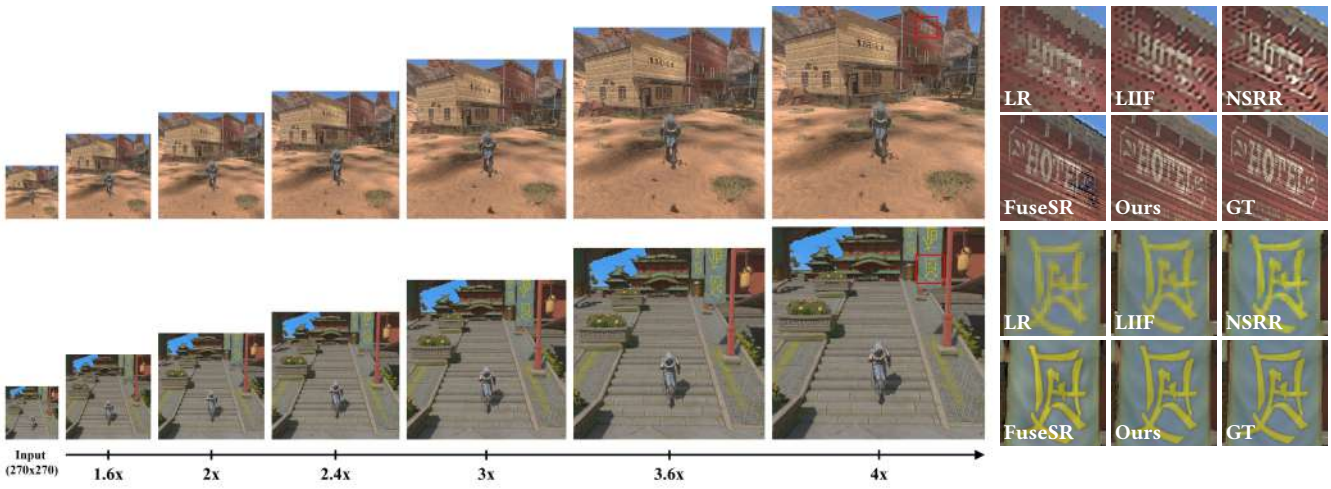

Figure 1: We propose a neural framework for arbitrary-scale super-resolution of rendering contents. Providing a single trained model, our framework is able to generate a new frame at an arbitrary resolution from an input low resolution (LR) frame. The insets show that our method produces sharp structures and textures that are very close to the ground truth (GT), and outperforms existing baselines: LIIF [Chen et al. 2021b], NSRR [Xiao et al. 2020], and FuseSR [Zhong et al. 2023].

ABSTRACT

As a prevailing tool for effectively reducing rendering costs in many graphical applications, frame super-resolution has seen important progress in recent years. However, most of prior works designed for rendering contents face a common limitation: once a model is trained, it can only afford a single fixed scale. In this paper, we attempt to eliminate this limitation by supporting arbitrary-scale

*Joint first authors.
†Corresponding author.

SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07…$15.00
https://doi.org/10.1145/3641519.3657439

super-resolution for a trained neural model. The key is a Fourier-based implicit neural representation which maps arbitrary and naturally coordinates in the high-resolution spatial domain to valid pixel values. By observing that high-resolution G-buffers possess similar spectrum to high-resolution rendered frames, we design a High-Frequency Fourier Mapping (HFFM) module to recover fine details from low-resolution inputs, without introducing noticeable artifacts. A Low-Frequency Residual Learning (LFRL) strategy is adopted to preserve low-frequency structures and ensure low biasedness caused by network inference. Moreover, different rendering contents are well separated by our spatial-temporal masks derived from G-buffers and motion vectors. Several light-weight designs to the neural network guarantee the real-time performance on a wide range of scenes.

CCS CONCEPTS

• **Computing methodologies** → **Ray tracing**.

KEYWORDS

Super-resolution, Fourier, Implicit neural representation, Real-time rendering

ACM Reference Format:
Haonan Zhang, Jie Guo, Jiawei Zhang, Haoyu Qin, Zesen Feng, Ming Yang, and Yanwen Guo. 2024. Deep Fourier-based Arbitrary-scale Super-resolution for Real-time Rendering. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641519.3657439

1 INTRODUCTION

The demand for high-resolution (HR), high-refresh-rate images and photo-realistic visual experiences is increasingly growing in various graphical scenarios. Moreover, with the emergence of hardware acceleration APIs, real-time ray tracing has been a popular trend in rendering content generation [Burgess 2020; Harada 2020; Sandy et al. 2018]. The heavy pixel workload brought by ray tracing presents a new and great challenge for existing rendering engines.

One prevailing and attracting scheme that has emerged to alleviate this problem consists in rendering the scene at a low-resolution (LR) and then adopting a super-sampling/super-resolution technique to achieve the desired high resolution. Several commercial solutions have been developed in this field, including Nvidia's DLSS series [Liu 2020], Intel's XeSS [Chowdhury et al. 2022], AMD's FSR [AMD 2023], and Epic Games' TAAU [Epic Games 2018]. These methods try to reuse neighboring pixel values by gathering samples across space and time, considering that most shading contents are spatially and temporally coherent [Scherzer et al. 2012; Yang et al. 2020]. Recently, deep learning (DL) based methods have become the mainstream. Like DLSS and XeSS, NSRR [Xiao et al. 2020] proposed by Xiao et al. is also a DL-based method which leverages a convolutional neural network (CNN) to achieve frame super-resolution up to 4×4 in real time. FuseSR [Zhong et al. 2023] proposed by Zhong et al. shows that utilizing low-cost HR G-buffers as additional input to the neural network can significantly improve the image quality. However, these DL-based methods can only perform a predefined fixed-scale super-resolution, limiting their potential in real-world applications.

In this paper, we devote to develop a single DL-based model that can achieve arbitrary-scale super-resolution for rendering contents. This is prohibitive for existing CNN-based methods (e.g. NSRR and FuseSR), since CNNs favor fixed-size representations. Inspired by recent progress in implicit neural representations for natural images [Chen et al. 2021b; Dupont et al. 2021], we also try to encode rendering contents in a continuous form via multilayer perceptron (MLP) that maps arbitrary coordinates to rendered pixel values. However, unlike arbitrary-scale super-resolution for natural images, our task handling rendering contents faces two new issues: 1) We have cheap and high-quality G-buffers that can provide full-resolution geometric information; 2) The network should be lightweight and runs in real time.

To address above issues, we propose a Fourier-based implicit neural representation, considering that rendered frames and G-buffers (such as depth, normal, and albedo) have quite different pixel values but possess similar spectrum in the Fourier space. Specifically, we divide the rendering contents into low-frequency parts and high-frequency parts and handle them separately. The high-frequency contents are identified by our spatial-temporal masks and undergo a High-Frequency Fourier Mapping (HFFM) module. The HFFM module utilizes Fourier mapping of latent features extracted from both the LR rendered frames and the HR G-buffers to help the network better predict high-frequency details. The low-frequency parts are handled by a Low-Frequency Residual Learning (LFRL) strategy to preserve low-frequency structures and ensure low biasedness. The spatial-temporal masks are generated heuristically and efficiently from G-buffers and motion vectors. Moreover, several special designs are involved in our pipeline to make it light-weight.

In summary, the main contributions of this paper are:

- an arbitrary-scale super-resolution framework for rendered frames based on implicit neural representations,
- an HFFM module for predicting high-frequency contents of rendered frames, which maps the neural features into the Fourier space and preserves details in frames with an arbitrary resolution,
- a residual learning strategy to ensure high-quality recovery of low-frequency contents,
- a spatial-temporal mask generation strategy to separate rendering contents into different groups, thus facilitating Fourier-based reconstruction.

2 RELATED WORK

Super-resolution/Super-sampling for real-time rendering. Frame super-sampling or super-resolution aims to recover an aliasing-free and HR rendered frame from its LR counterpart. In early methods, historical frames are resampled with the help of accurate motion vectors and some form of temporal reprojection operations [Yang et al. 2020, 2009]. Since shading and visibility changes between adjacent frames easily cause artifacts, some heuristic neighborhood clamping strategies have been proposed to correct the historical samples [Karis 2014; Salvi 2016]. Despite the high efficiency, these

heuristic methods are prone to errors and thus easily incur ghosting and other artifacts including loss of details, temporal lag and residual noise in the resultant frames.

Recently, convolutional neural networks offer a promising alternative for frame super-sampling [Chowdhury et al. 2022; Liu 2020; Xiao et al. 2020; Yang et al. 2023; Zhong et al. 2023]. Xiao et al. [2020] proposed NSRR which combines dense motion vectors and depth with a typical encoder-decoder network. This network takes five consecutive frames as input and predicts up to 4× upsampled frames with high fidelity and temporal stability. Yang et al. [2023] proposed MNSS, an efficient neural super-sampling framework that can run on mobile devices. Zhong et al. [2023] suggested that using HR G-buffers can significantly improve the image quality. In industry, Nvidia has released DLSS [Liu 2020] that upsamples LR frames with neural networks in real time. Intel has also developed a learning-based frame upsampling technique named XeSS [Chowdhury et al. 2022] which also achieves promising results on a wide range of scenes. Indeed, some of these techniques either have undisclosed technical details or require training specific models for each super-resolution scale factor, which severely limits their practical application.

Arbitrary-scale super-resolution. While less explored in rendering, arbitrary-scale super-resolution has recently become a hot topic in computer vision [Chen et al. 2023, 2021b; Hu et al. 2019; Son and Lee 2021]. As a pioneer work, MetaSR [Hu et al. 2019] chooses to generate convolution kernels based on coordinates and scale factor information, thereby proposing the first method for arbitrary-scale super-resolution. Recent works suggested use implicit neural representations to achieve arbitrary-scale super-resolution. Chen et al. [2023] propose LIT and CLIT which integrate the attention mechanism and frequency encoding technique into a local implicit image function. The utilization of Fourier information and dominant-frequency estimator in Local Texture Estimator (LTE) [Lee and Jin 2022] for implicit feature representation has effectively improved the learning ability and inference efficiency of implicit neural representations. Inspired by this, we also resort to Fourier-based neural representations to enhance image details. However, the usage of Fourier information is quite different. In particular, our task has cheap G-buffers that can largely exploited the Fourier information in complex scenes.

Implicit neural representations. Implicit neural representations, which typically parameterize signals using MLP, have been shown to offer competitive advances over explicit representations. In recent years, they have been successfully applied to model complex signals like images [Dupont et al. 2021], videos [Chen et al. 2021a], 3D shapes [Liu et al. 2022; Park et al. 2019], 3D scenes [Jiang et al. 2020], textures [Oechsle et al. 2019] and radiance fields [Mildenhall et al. 2020]. These neural representations are differentiable and generally more compact compared to canonical representations [Sitzmann et al. 2020]. They allow discrete signals to be treated as continuous over bounded domains, since they naturally support smooth interpolations to unseen coordinates. Our approach applies implicit neural representations to render contents and utilizes HR G-buffers for extracting frequency information of frames.

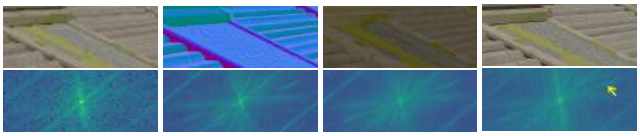

Figure 2: Demonstration of an LR frame $\mathbf{i}_t$, and its HR G-buffers: $\mathbf{N}_t$ and $\mathbf{A}_t$. These G-buffers possess similar spectrum to the corresponding HR frame $\bar{\mathbf{I}}_t$. The yellow arrow indicates that some high-frequency contents which do not exist in $\mathbf{i}_t$ can be recovered from HR G-buffers. Please zoom in to see more detailed information.

3 METHOD

In this section, we detail our neural pipeline to support arbitrary-scale super-resolution for rendering contents, starting with the description of the motivation behind the problem.

3.1 Motivation

Modern displays have many different screen resolutions: 1920×1080, 2560 × 1440, 1440 × 900, 1366 × 768, et al. When applying some DL-based super-resolution techniques to rendering applications running on devices equipped with these displays, a great challenge occurs: one has to train quite a few models to handle each scaling factor separately. Although we can achieve non-integer scaling in super-resolution by additional operations such as bilinear scaling to modify the network input or output size when handling dynamic resolutions, the resulting repetitive computational cost limits the practical application of these methods.

Unlike natural images, rendering contents contain cheap G-buffers that can significantly improve the quality of reconstructed frames [Guo et al. 2021; Zhong et al. 2023]. Clearly, these G-buffers have quite different pixel values and data distributions than rendered frames, but we observe that they show similar spectrum when converted to the Fourier space, as demonstrated in Fig. 2. As seen, the albedo $\mathbf{A}_t$ and normal $\mathbf{N}_t$ have very similar spectral lines to the HR frame $\bar{\mathbf{I}}_t$, while the spectrum of the LR frame $\mathbf{i}_t$ lacks many details. This inspires us to incorporate Fourier information of HR G-buffers into the implicit neural representation.

3.2 Problem setup and overview

We focus on arbitrary-scale super-resolution for rendering contents. The input to this task includes the scaling factor r, the LR rendered frame $\mathbf{i}_t$ at the current time step t, its LR and HR G-buffers $\mathbf{g}_t, \mathbf{G}_t$, along with some historical frames $\{\mathbf{I}_{t'}\}(t' < t)$ and the corresponding HR G-buffers $\{\mathbf{G}_{t'}\}$. Our method predicts a super-sampled frame $\mathbf{I}_t$ satisfying the scaling factor r. This process can be mathmatically formulated as

$$\mathbf{I}_t = \Theta_{ASSR}(\mathbf{i}_t, \mathbf{g}_t, \mathbf{G}_t, \{\mathbf{I}_{t'}\}, \{\mathbf{G}_{t'}\}, r). \tag{1}$$

Our whole pipeline Θ_{ASSR} consists of three different groups of G-buffers: $\mathbf{G}_t = \{\mathbf{A}_t, \mathbf{N}_t, \mathbf{D}_t, \mathbf{MV}_t, \mathbf{B}_t\}$, $\mathbf{g}_t = \{\mathbf{a}_t, \mathbf{n}_t, \mathbf{d}_t, \mathbf{b}_t\}$, and $\mathbf{G}_{t'} = \{\mathbf{A}_{t'}, \mathbf{N}_{t'}, \mathbf{D}_{t'}\}$. $\mathbf{A}/\mathbf{a}$, $\mathbf{N}/\mathbf{n}$, $\mathbf{D}/\mathbf{d}$, $\mathbf{B}/\mathbf{b}$, and $\mathbf{MV}$ denote albedo, normal, depth, pre-integrated BRDF information [Zhuang et al. 2021], and motion vector, respectively. Note that these G-buffers are easy and cheap to obtain in modern rendering engines. Inspired by [Zhuang

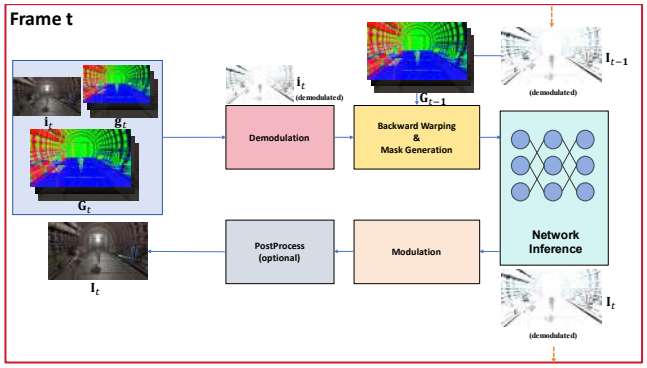

Figure 3: Illustration of our whole pipeline for arbitrary-scale super-resolution.

et al. 2021] and [Zhong et al. 2023], we choose the pre-integrated BRDF information to perform demodulation before feeding $\mathbf{i}_t$ into the network and perform modulation on the predicted result. For efficiency, we currently set $t' = t - 1$, which means we only use one historical frame in our pipeline.

As shown in Fig. 3, our pipeline works in a temporally recursive manner: the inference result and G-buffers of the previous frame $\mathbf{I}_{t-1}, \mathbf{G}_{t-1}$ are warped by the motion vector $\mathbf{MV}_t$ and fed to the process of the current timestamp. The current frame $\mathbf{i}_t$ is demodulated while $\mathbf{g}_t, \mathbf{G}_t$ as well as the previous warped G-buffers will also be used to generate spatial-temporal masks. After network inference, the super-sampled new frame is modulated and is applied by other post-process steps like tone mapping. The detailed architecture of our network is shown in Fig. 4, which mainly consists of a High-Frequency Fourier Mapping (HFFM) module, a Low-Frequency Residual Learning (LFRL) strategy, and several feature extraction modules with light-weight designs.

3.3 High-frequency Fourier mapping

To support arbitrary-scale super-resolution, we design an implicit neural representation for rendering contents. Given an arbitrary queried coordinate of the output HR frame, this implicit neural representation, realized by fully connected MLP, is expected to predict a plausible pixel value for any scene.

A major limitation of MLP-based implicit neural representations is their difficulty in learning high-frequency functions, due to the existance of spectral bias [Basri et al. 2020; Rahaman et al. 2019]. Existing works addressing this issue resort to some kind of positional encoding strategies [Mildenhall et al. 2020]. In our task, we have cheap HR G-buffers which can compensate the missing details in the LR frames. Based on the above analysis, we try to insert high-frequency G-buffers' information into the implicit neural representation, via Fourier analysis. Specifically, we design the HFFM module in our pipeline to combine the implicit neural representation with Fourier information stemming from HR G-buffers, thus enhancing the high-frequency details of the super-resolution results.

The goal of the HFFM module is to generate a high-frequency feature code $\mathbf{F}_H$ for a pixel coordinate x in the HR spatial domain, based on the feature maps extracted from the input LR frame $\mathbf{i}$, HR G-buffers $\mathbf{G}$, and scaling factor r:

$$\mathbf{F}_H(x) = \mathbf{F}_\mathbf{i}(x^*) \otimes \begin{bmatrix} \cos[\mathbf{Dot}(\mathbf{F}_\mathbf{G}(x), \delta)\pi + \mathbf{P}(r)] \\ \sin[\mathbf{Dot}(\mathbf{F}_\mathbf{G}(x), \delta)\pi + \mathbf{P}(r)] \end{bmatrix}. \quad (2)$$

Here, $\otimes$ denotes the element-wise multiply operation, x^* represents the coordinate of the nearest pixel in the LR spatial domain from x, and $\delta = x - x^*$ is the relative coordinate indicating the distance between the LR spatial position x^* and the pixel position x in the HR frame. For brevity, we omit the current time step t. Intuitively, $\mathbf{F}_\mathbf{i}$, $\mathbf{F}_\mathbf{G}$, and $\mathbf{P}$ respectively encode the amplitude, frequency, and phase of the feature code $\mathbf{F}_H$. The details are explained as follows.

(1) The amplitude component $\mathbf{F}_\mathbf{i} \in \mathbb{R}^C$ is obtained by querying the nearest (Euclidean distance) feature code from the feature maps extracted from the LR frame $\mathbf{i}$. We omit features from G-buffers, considering that G-buffers will introduce bias to the final shading through the amplitude component. Extracting and computing amplitude features from the LR frame alone enables the implicit features to align with the continuous representation in the image space.

(2) The frequency component $\mathbf{F}_\mathbf{G} \in \mathbb{R}^{\frac{C}{2} \times 2}$ is obtained by querying the feature code directly from the feature maps extracted from HR G-buffers. Since the HR G-buffers have the same resolution to the output, no upsampling operation is involved here. This largely preserves the details in the G-buffers. These HR G-buffers are expected to provide complementary frequency information in our high-frequency signal prediction task.

(3) The phase component $\mathbf{P} \in \mathbb{R}^{\frac{C}{2}}$ is generated by mapping the scaling factor r to a $\frac{C}{2}$-dimensional vector, with a one-layer perceptron. The phase component helps to locate the edge of features when the scaling factor changes. When the scaling factor changes, the locations of feature boundaries will also change within a small range and the network learns the scale-related information from the phase component.

Once obtained the high-frequency feature code $\mathbf{F}_H$, we feed it to a 4-layer MLP and an additional convolutional layer to produce the resultant pixel value. $C = 64$ represents the number of neurons per layer.

HFFM employs learnable Fourier feature mapping (as shown in Eq. (2)) combined with an MLP to approximate the dominant Fourier frequencies and the corresponding Fourier coefficients in the HR images. This differs from predicting pixel values through coordinates and positional encoding like that in NeRF [Mildenhall et al. 2020]. We need to identify the effective information for the super-resolution task in the network inputs and categorize different encodings into the constituent components of Eq. (2). In other words, we convert coordinates into the Fourier domain before feeding them to the MLP in a data-driven manner.

3.4 Spatial-temporal masks

To better handle high-frequency details, we propose spatial-temporal masks to explicitly identify potential high-frequency regions in the final output. These masks, also fed into the network, serve as a good guidance for recovering details.

The spatial masks are generally generated by computing the difference between HR G-buffers (depth $\mathbf{D}_t$, normal $\mathbf{N}_t$, and albedo

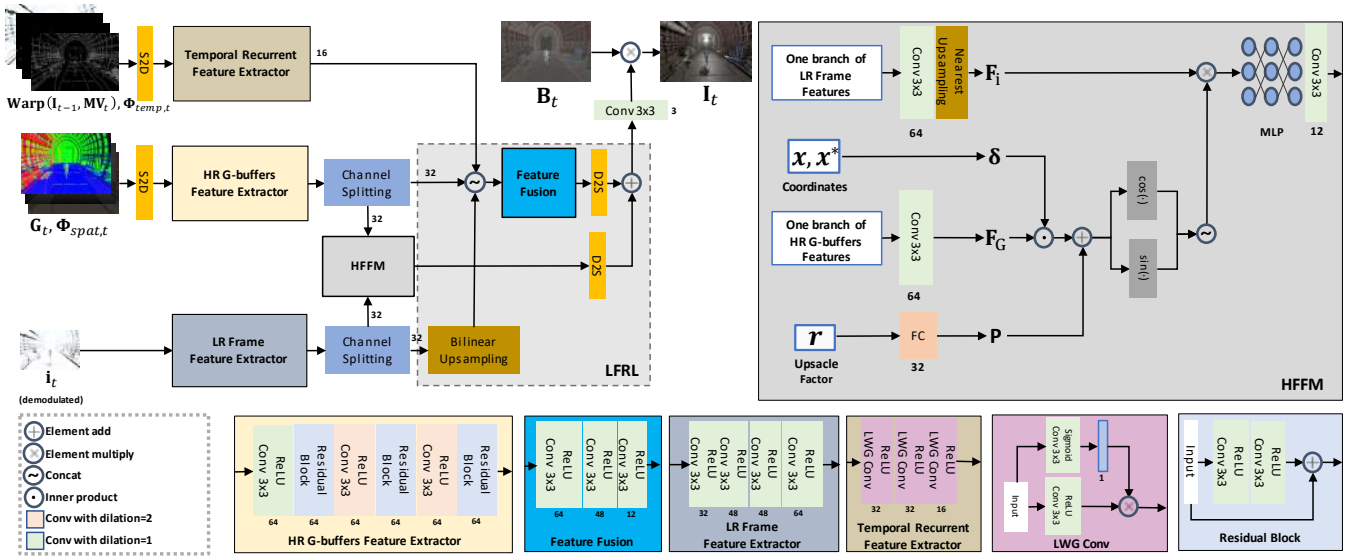

Figure 4: Network architecture in our pipeline. The top left corner shows an outline of our network architecture and the top right corner presents the details of the HFFM module. Some other feature extraction modules are shown in the lower part of the figure. In particular, channel splitting represents splitting the features into two branches along the channel dimension. The LWG layer [Yi et al. 2020] enables a learnable feature selection mechanism. The numbers next to each network layer represent the output channels at corresponding layers. Note that the LR rendered image i_t and the recurrent previous result I_{t-1} are both demodulated by pre-integrated BRDF information [Zhuang et al. 2021]. More details about channel splitting [Guo et al. 2022], S2D and D2S [Shi et al. 2016] operations are shown in 3.6.

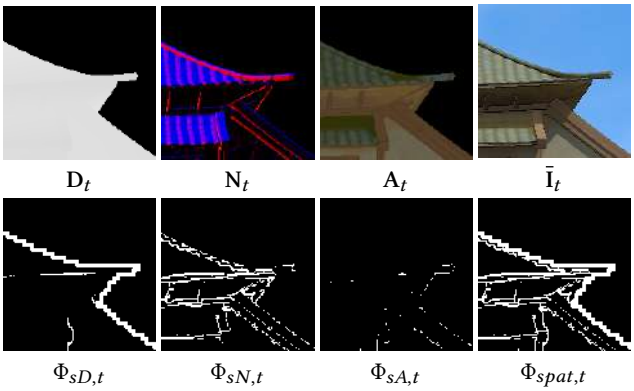

Figure 5: Demonstration of spatial masks.

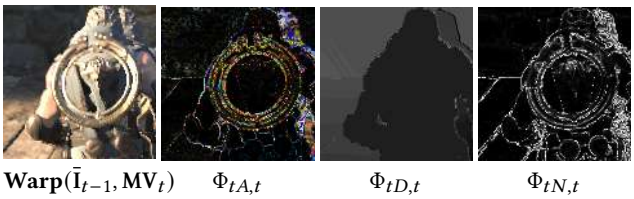

Figure 6: Demonstration of temporal masks.

$\mathbf{A}_t$) and bilinearly upsampled LR G-buffers (depth $\mathbf{d}_t$, normal $\mathbf{n}_t$, and albedo $\mathbf{a}_t$). Specifically, we identify pixels according to the following conditions:

$$\Phi_{sD,t} = \mathbf{H}(|\mathbf{D}_t - \mathbf{Up}(\mathbf{d}_t)| - \phi_D) \quad (3)$$

$$\Phi_{sA,t} = \mathbf{H}(|\mathbf{A}_t - \mathbf{Up}(\mathbf{a}_t)| - \phi_A) \quad (4)$$

$$\Phi_{sN,t} = \mathbf{H}(\phi_N - \mathbf{Dot}(\mathbf{N}_t, \mathbf{Up}(\mathbf{n}_t)) \quad (5)$$

where ϕ_D, ϕ_A, and ϕ_N are predefined thresholds. $\mathbf{H}$ is the Heaviside step function to ensure that all spatial masks are binary masks. For the normal buffer, we consider measuring the difference by the angle between the normal vectors of two pixels, where a small angle indicates the high probability of two pixels having similar shading. Therefore, we calculate the cosine value between the HR normal buffer $\mathbf{N}_t$ and the upsampled LR normal buffer $\mathbf{Up}(\mathbf{n}_t)$. The final spatial mask $\Phi_{spat,t}$ is defined as

$$\Phi_{spat,t} = \Phi_{sD,t} \cup \Phi_{sA,t} \cup \Phi_{sN,t}. \quad (6)$$

The binary design of the spatial mask can identify pixels that cover high-frequency contents and is convenient for loss calculation. Fig. 5 shows an example of the marked pixels by different conditions.

Inspired by TAA [Tatarchuk et al. 2014] and ExtraNet [Guo et al. 2021], we also reuse temporal information from historical frames. After warping, some pixels in the historical frames become invalid. To identify these invalid pixels, we generate temporal masks, via the G-buffers of the current frame ($\mathbf{D}_t, \mathbf{A}_t, \mathbf{N}_t$) and the previous

frame $(\mathbf{D}_{t-1}, \mathbf{A}_{t-1}, \mathbf{N}_{t-1})$:

$$\Phi_{tD,t} = |\mathbf{D}_t - \mathbf{Warp}(\mathbf{D}_{t-1}, \mathbf{MV}_t)| \quad (7)$$

$$\Phi_{tA,t} = |\mathbf{A}_t - \mathbf{Warp}(\mathbf{A}_{t-1}, \mathbf{MV}_t)| \quad (8)$$

$$\Phi_{tN,t} = \frac{1 - \mathbf{Dot}(\mathbf{N}_t, \mathbf{Warp}(\mathbf{N}_{t-1}, \mathbf{MV}_t))}{2}. \quad (9)$$

The final temporal mask is defined as

$$\Phi_{temp,t} = [\Phi_{tD,t}, \Phi_{tA,t}, \Phi_{tN,t}] \quad (10)$$

where $[\cdot, \cdot]$ represents the channel-wise concatenation, $\mathbf{MV}_t$ is the motion vector generated by the render engine, and $\mathbf{Warp}(\cdot, \cdot)$ is the backward warping function which is also used by almost all the common temporal reconstruction methods [Scherzer et al. 2012]. Unlike the spatial mask $\Phi_{spat,t}$, the temporal mask $\Phi_{temp,t}$ is defined as soft masks: $\Phi_{temp,t}$ has five channels with continuous value in $[0, 1]^5$. The soft mask can largely reuse temporal information from historical frames. Fig. 6 shows an example of the temporal mask.

3.5 Low-frequency residual learning

To ensure low biasedness caused by network inference, we design an LFRL strategy in our network. The key idea is to preserve global structural information in the LR frames as much as possible, without being negatively impacted by the HR G-buffers. LFRL leverages residual learning to minimize the bias introduced by features from G-buffers. It also enriches high-frequency components in residuals and stabilizes convergence.

Instead of using the bilinear upsampled frame of $\mathbf{i}_t$ directly for residual learning, we let $\mathbf{i}_t$ pass through an LR Frame Feature Extractor to remove aliasing and noise in the original LR frames. The temporal features from Temporal Recurrent Feature Extractor can help the result maintain temporal stability. Moreover, HR G-buffers contain both high-frequency and low-frequency information. Therefore, we include the corresponding features as part of the input.

We concatenate these different features together via a Feature Fusion module, resulting in the global low-frequency feature $\mathbf{F}_L$. After the D2S operation, this part is added to the high-frequency component generated by HFFM and finally passed through a convolutional layer to obtain the prediction result.

3.6 Lightweight designs

To improve the performance of network inference, we use space-to-depth (S2D) [Shi et al. 2016] operation for HR data, like $\mathbf{G}_t$ and $\mathbf{I}_{t-1}$. The output is mapped to the target resolution using a depth-to-space (D2S). This helps the network extract features and make super-resolution predictions at $\frac{1}{2} \times \frac{1}{2}$ times the target resolution.

To reduce the memory and computational overhead further, we use the operation of channel splitting [Guo et al. 2022], which splits the features into two branches along the channel dimension, dividing them equally. The output features of LR Frame Feature Extractor will be divided into two branches by channel splitting: one serves as the amplitude component of HFFM and the other is fed into the Feature Fusion module after bilinear upsampling. The output features of HR G-buffers Feature Extractor are also split into two branches that are fed into HFFM as frequency component and the Feature Fusion module, respectively. The different dilation rates allows us to expand the receptive field while maintaining the spatial relationships within the feature maps.

The heavy operations such as unfold operation in LIIF [Chen et al. 2021b] to enhance learning capability are not feasible for our pipeline. Therefore, besides reducing the number of neurons in MLP, we also incorporate 3×3 convolutions in HFFM which increases the receptive field without introducing significant computational overhead. We also add two convolutional layers at the end of HFFM and the whole network which can alleviate discontinuity in predictions.

4 EXPERIMENTS

4.1 Datasets and implementation details

Datasets. Our training and test datasets are generated using a modified version of Unreal Engine 4. The training dataset consists of four scenes: Bunker (**BK**), RedwoodForest (**RF**), MedievalDocks (**MD**), and WesternTown (**WT**). Besides the above four scenes, the test dataset also includes another scene, EasternVillage (**EV**), to evaluate the generalization ability of our method. To support arbitrary-scale super-resolution, we render the scenes with three different scaling factors: 2×2, 3×3, and 4×4, besides the original LR frames. Please refer to our supplemental material for more details.

Loss function. The training loss $\mathcal{L}$ of our network is a weighted combination and the details are provided in the supplemental material.

Implementation. Unlike previous arbitrary-scale super-resolution studies that downsampling data with a uniform distribution of the scale factors during training, our network randomly selects from $\{2, 3, 4\}$ each iteration and reads the corresponding LR data for training. The network is implemented and trained using the PyTorch framework [Paszke et al. 2019]. Adam optimizer is used for optimization and its parameters (β_1, β_2) are set to $(0.9, 0.999)$. We use 8-frame clips and random patches with size 240×240 for training. The initial learning rate is 0.0001 and decays half every 20 epochs. All tests are run on a PC equipped with an NVIDIA RTX 3090 GPU.

4.2 Comparison

To demonstrate the effectiveness of our method, we compare it against some state-of-the-art methods. For 4×4 super-resolution, we choose NSRR [Xiao et al. 2020], LIIF [Chen et al. 2021b] and FuseSR [Zhong et al. 2023] as the competitors. Note that both our model and LIIF are trained on the whole training dataset, while NSRR and FuseSR are only trained on the 4×4 data since both methods do not support arbitrary scaling factors. For 3×3 super-resolution, we compare our method to LIIF, NSRR, and FSR [AMD 2023]. FSR is open-source technique widely used in the gaming industry.

Quantitative comparison. To quantitatively analyze these results, we choose signal-to-noise ratio (PSNR), structural similarity index (SSIM) [Wang et al. 2004] and LPIPS [Zhang et al. 2018] as the error metrics. We evaluate five scenes and the quantitative results are reported in Table 1 and Table 2, respectively. Our method generally

Table 1: Quantitative comparison in terms of PSNR, SSIM and LPIPS on all scenes for the 4×4 scaling factor. The best scores are shown in bold.

	Method	BK	RF	MD	WT	EV
PSNR(dB)↑	LIIF	23.63	14.23	19.44	22.92	23.52
	NSRR	23.70	13.66	19.19	22.26	23.10
	FuseSR	25.20	17.50	21.57	24.77	24.72
	Ours	**25.62**	**18.24**	**22.94**	**26.02**	**25.57**
SSIM↑	LIIF	0.7542	0.2352	0.6435	0.6371	0.7069
	NSRR	0.7576	0.2448	0.6472	0.6326	0.6988
	FuseSR	**0.8026**	0.5163	0.7843	**0.7854**	**0.8083**
	Ours	0.7720	**0.5729**	**0.8051**	0.7594	0.8013
LPIPS↓	LIIF	0.4312	0.6637	0.4480	0.4757	0.3973
	NSRR	0.4256	0.6264	0.4272	0.4649	0.3958
	FuseSR	0.2579	0.3710	0.2520	**0.2251**	0.2199
	Ours	**0.2494**	**0.3294**	**0.2256**	0.2270	**0.1934**

Table 2: Quantitative comparison in terms of PSNR, SSIM and LPIPS on all scenes for the 3×3 scaling factor. The best scores are shown in bold.

	Method	BK	RF	MD	WT	EV
PSNR(dB)↑	LIIF	24.67	14.58	20.42	23.95	24.55
	NSRR	26.64	15.55	21.28	25.38	25.16
	FSR	**27.07**	**19.16**	22.41	**26.60**	25.75
	Ours	26.37	18.49	**23.64**	26.58	**26.24**
SSIM↑	LIIF	0.7979	0.2946	0.7107	0.6927	0.7576
	NSRR	**0.8287**	0.3425	0.7490	0.7335	0.7688
	FSR	0.8219	0.5595	0.7000	0.7472	0.7769
	Ours	0.7999	**0.5910**	**0.8202**	**0.7760**	**0.8215**
LPIPS↓	LIIF	0.3494	0.5958	0.3797	0.3998	0.3169
	NSRR	0.3440	0.5606	0.3511	0.3799	0.3290
	FSR	0.2990	0.3660	0.3364	0.3138	0.2950
	Ours	**0.2301**	**0.3200**	**0.2116**	**0.2155**	**0.1779**

outperforms its competitors on either 4×4 super-resolution or 3×3 super-resolution. We should emphasize again that NSRR, FuseSR, and FSR require training separate models for each scaling factor. Additionally, we evaluate the capability of our method for non-integer and out-of-training-distribution scales and the details are shown in the supplemental material.

Qualitative comparison. In Fig. 11, we show the qualitative comparisons of different methods for 4×4 super-resolution and Fig. 12 provides the qualitative comparison of 3×3 super-resolution on **BK**, **RF** and **MD** scenes. Our method faithfully produces high-frequency details such as sharp edges and complex textures that cannot be well reconstructed by FSR [AMD 2023]. Additionally, we provide the qualitative comparisons against FuseSR for 2×2 super-resolution in the supplemental material. LIIF is originally designed for natural images. Without G-buffers, it tends to genrate overly blur results. NSRR and FSR also fail to recover high-frequency details, since they only relies on LR G-buffers. Equipped with HR G-buffers, FuseSR tends to generate super-sampled results better than LIIF and NSRR. However, it generates obvious artifacts along high-frequency boundaries, since the H-Net in FuseSR does not always guarantee

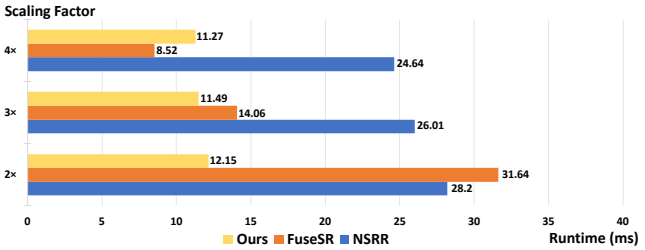

Figure 7: Comparison of runtime of network inference between our method and other baseline methods at 1080P resolution. LIIF [Chen et al. 2021b] is not shown in the figure as its high inference cost ($\gg$ 100 ms).

the alignment between HR G-buffers and LR inputs. Moreover, FuseSR produces color distortions while our method with the residual learning strategy avoids this issue.

Although our model is only trained on data with 2×2 to 4×4 scaling factors, it can handle other scaling factors beyond that scope. To show this, we test our model on 6×6 and 8×8 super-resolution. The results are provided in Fig. 13. Among the baseline methods, only LIIF [Chen et al. 2021b] has the same capability of arbitrary-scale super-resolution, and our method significantly outperforms LIIF in these cases. Our model can also support non-integral scaling factors, as shown in Fig. 1.

4.3 Runtime performance analysis

We display runtime comparison of our method against other baseline methods with different scaling factors which target at a 1080P resolution in Fig. 7. NVIDIA TensorRT is used for acceleration and all neural networks are evaluated in FP16. LIIF [Chen et al. 2021b] is not shown in the figure as its inference cost is much higher than 100 ms. For the 4×4 scaling factor, our method (11.27 ms) is slightly slower than FuseSR (8.52 ms) while we show better performance than other baseline methods at 2×2 and 3×3. It is also worth noting that the performance of our method can be significantly improved with delicate engineering, such as quantizing the MLPs to INT8 or compute shaders.

4.4 Ablation study

Validation of spatial-temporal masks. Fig. 8 shows the effects of spatial-temporal masks on the final reconstructed frames. The spatial masks are used to identify different frequencies in the frame. Without them, the reconstructed frames may lose details. The temporal masks can identify the invalid areas from the warped previous frames. While removing temporal masks improves PSNR scores slightly (refer to our supplemental material), we find that this step is beneficial for the reconstruction of thin objects and edges. This is revealed in Fig. 8.

Validation of HFFM and LFRL. In Fig. 9, we validate the effectiveness of HFFM and LFRL, and compare super-sampled frames both visually and quantitatively. Removing LFRL introduces aliasing artifacts in the LR frame, leading to holes and ghosting artifacts. Additionally, directly predicting the image using MLP, instead of

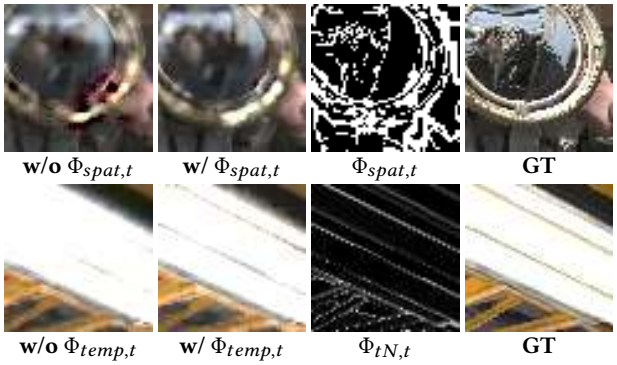

Figure 8: Comparison between models trained without/with $\Phi_{spat,t}/\Phi_{temp,t}$ on the BK scene with a 4×4 scaling factor. The third column shows the spatial mask and the temporal mask, respectively.

Figure 9: Comparison between models trained without/with HFFM/LFRL on the MD scene with a 4×4 scaling factor. The metrics of PSNR and SSIM of the whole frames are shown below the figure.

Table 3: Quantitative comparison in terms of PSNR, SSIM and LPIPS without different components of HFFM. The metrics are averaged on the MD and RF scenes and the best scores are shown in bold.

	Ours	*w/o A*	*w/o F*	*w/o P*
PSNR(dB)↑	**20.33**	20.13	20.24	20.17
SSIM↑	**0.6980**	0.6934	0.6971	0.6881
LPIPS↓	**0.2747**	0.2807	0.2772	0.2841

relying on mapping into the Fourier space, lowers the ability of our network to capture high-frequency details.

Validation of different components in HFFM. To validate the importance of each component in Eq (2), we retrain the following models without the amplitude component $\mathbf{F_i}$ (*w/o A*), the frequency component $\mathbf{F_G}$ (*w/o F*) or the phase component $\mathbf{P}$ (*w/o P*). The quantitative comparison is shown in Table 3 and the visual comparison is shown in Fig. 10. Clearly, these ablated models achieve low accuracy as compared with our complete model.

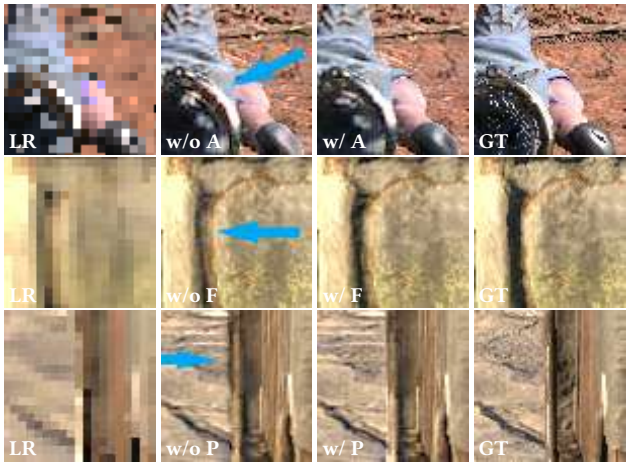

Figure 10: Comparison between models trained without/with the component of amptitude/frequency/phase in HFFM with a 4×4 scaling factor. Blue arrows highlight the differences.

5 CONCLUSION AND DISCUSSION

Limitation. As a common limitation for many super-resolution methods, for effects such as fog and particles, we cannot obtain their information from G-buffers. Therefore, the frames with such special effects cannot be effectively processed by our method. Moreover, because of the perceptual loss used for training, our method may occasionally produce pattern artifacts.

Couclusion. In this paper, we propose a real-time framework for arbitrary-scale super-resolution of rendered frames. The basic idea in our method is using a Fourier-based implicit neural representation to encode rendering contents. We design an HFFM module to reconstruct high-frequency contents in the Fourier space. We also employ a low-frequency residual learning strategy to ensure high-quality recovery of low-frequency contents. Spatial-temporal masks are involved to improve the performance of predicted results. Several light-weight designs help the model achieve real-time inference speed. To the best of our knowledge, the proposed method is the first to support arbitrary-scale super-resolution in the context of rendering. The investigation of our method with 6×6 and 8×8 scaling factor is exploratory in nature and still remains significant room for improvement in handling subpixel details. We did not employ jittered sequences, which is also an interesting future research direction. We hope this work could promote new research interests in rendered frame super-resolution.

REFERENCES

AMD. 2023. AMD FSR 3 Now Available. https://www.amd.com/en/technologies/fidelityfx-super-resolution/.

Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. 2020. Frequency Bias in Neural Networks for Input of Non-Uniform Density. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 64, 10 pages.

John Burgess. 2020. RTX on the NVIDIA Turing GPU. *IEEE Micro* 40, 2 (2020), 36–44.

Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser-Nam Lim, and Abhinav Shrivastava. 2021a. NeRV: Neural Representations for Videos. In *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (Eds.). https://openreview.net/forum?id=BbikqBWZTGB

Hao-Wei Chen, Yu-Syuan Xu, Min-Fong Hong, Yi-Min Tsai, Hsien-Kai Kuo, and Chun-Yi Lee. 2023. Cascaded Local Implicit Transformer for Arbitrary-Scale Super-Resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*. IEEE, 18257–18267. https://doi.org/10.1109/CVPR52729.2023.01751

Yinbo Chen, Sifei Liu, and Xiaolong Wang. 2021b. Learning Continuous Image Representation With Local Implicit Image Function. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 8628–8638. https://doi.org/10.1109/CVPR46437.2021.00852

Hisham Chowdhury, Kawiak, Rense Robert, de Boer, Gabriel Ferreira, and Lucas Xavier. 2022. Intel XeSS – an AI based Super Sampling solution for Real-time Rendering. In *Game Developers Conference*.

Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. 2021. COIN: COmpression with Implicit Neural representations. https://arxiv.org/abs/2103.03123

Epic Games. 2018. Unreal Engine 4.19: Screen Percentage with Temporal Upsample. https://docs.unrealengine.com/en-US/Engine/Rendering/ScreenPercentage/index.html. Accessed in August 2019.

Jie Guo, Xihao Fu, Liqiang Lin, Hengjun Ma, Yanwen Guo, Shiqiu Liu, and Ling-Qi Yan. 2021. ExtraNet: real-time extrapolated rendering for low-latency temporal supersampling. *ACM Trans. Graph.* 40, 6 (2021), 278:1–278:16. https://doi.org/10.1145/3478513.3480531

Yu-Xiao Guo, Guojun Chen, Yue Dong, and Xin Tong. 2022. Classifier Guided Temporal Supersampling for Real-time Rendering. *Comput. Graph. Forum* 41, 7 (2022), 237–246. https://doi.org/10.1111/CGF.14672

Takahiro Harada. 2020. Hardware-Accelerated Ray Tracing in AMD Radeon ProRender 2.0. https://gpuopen.com/learn/radeon-prorender-2-0/.

Xuecai Hu, Haoyuan Mu, Xiangyu Zhang, Zilei Wang, Tieniu Tan, and Jian Sun. 2019. Meta-SR: A Magnification-Arbitrary Network for Super-Resolution. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 1575–1584. https://doi.org/10.1109/CVPR.2019.00167

Chiyu Max Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. 2020. Local Implicit Grid Representations for 3D Scenes. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Brian Karis. 2014. High-Quality Temporal Supersampling. SIGGRAPH 2014 Advances in Real-Time Rendering in Games course.

Jaewon Lee and Kyong Hwan Jin. 2022. Local Texture Estimator for Implicit Representation Function. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 1919–1928. https://doi.org/10.1109/CVPR52688.2022.00197

Edward Liu. 2020. DLSS 2.0 - Image Reconstruction for Real-Time Rendering with Deep learning. In *Game Developers Conference*.

Hsueh-Ti Derek Liu, Francis Williams, Alec Jacobson, Sanja Fidler, and Or Litany. 2022. Learning Smooth Neural Functions via Lipschitz Regularization. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 31, 13 pages.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12346)*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer, 405–421. https://doi.org/10.1007/978-3-030-58452-8_24

Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. 2019. Texture Fields: Learning Texture Representations in Function Space. In *Proceedings IEEE International Conf. on Computer Vision (ICCV)*.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 8024–8035. https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron C. Courville. 2019. On the Spectral Bias of Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 5301–5310. http://proceedings.mlr.press/v97/rahaman19a.html

Marco Salvi. 2016. An excursion in temporal supersampling. Game Developer's Conference (GDC) 2016.

Matt Sandy, Johan Andersson, and Colin Barré-Brisebois. 2018. DirectX: Evolving Microsoft's Graphics Platform. Game Developers Conference 2018.

Daniel Scherzer, Lei Yang, Oliver Mattausch, Diego Nehab, Pedro V. Sander, Michael Wimmer, and Elmar Eisemann. 2012. Temporal Coherence Methods in Real-Time Rendering. *Comput. Graph. Forum* 31, 8 (dec 2012), 2378–2408.

Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 1874–1883. https://doi.org/10.1109/CVPR.2016.207

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. 2020. Implicit Neural Representations with Periodic Activation Functions. In *Proc. NeurIPS*.

Sanghyun Son and Kyoung Mu Lee. 2021. SRWarp: Generalized Image Super-Resolution under Arbitrary Transformation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 7782–7791. https://doi.org/10.1109/CVPR46437.2021.00769

Natasha Tatarchuk, Brian Karis, Michal Drobot, Nicolas Schulz, Jerome Charles, and Theodor Mader. 2014. Advances in real-time rendering in games, part I. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH '14, Vancouver, Canada, August 10-14, 2014, Courses*. ACM, 10:1. https://doi.org/10.1145/2614028.2615455

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

Lei Xiao, Salah Nouri, Matt Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. 2020. Neural Supersampling for Real-Time Rendering. *ACM Trans. Graph.* 39, 4, Article 142 (July 2020), 12 pages.

Lei Yang, Shiqiu Liu, and Marco Salvi. 2020. A Survey of Temporal Antialiasing Techniques. *Computer Graphics Forum* 39, 2 (2020), 607–621.

Lei Yang, Diego Nehab, Pedro V. Sander, Pitchaya Sitthi-amorn, Jason Lawrence, and Hugues Hoppe. 2009. Amortized Supersampling. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 1–12.

Sipeng Yang, Yunlu Zhao, Yuzhe Luo, He Wang, Hongyu Sun, Chen Li, Binghuang Cai, and Xiaogang Jin. 2023. MNSS: Neural Supersampling Framework for Real-Time Rendering on Mobile Devices. *IEEE Transactions on Visualization and Computer Graphics* (2023), 1–14. https://doi.org/10.1109/TVCG.2023.3259141

Zili Yi, Qiang Tang, Shekoofeh Azizi, Daesik Jang, and Zhan Xu. 2020. Contextual Residual Aggregation for Ultra High-Resolution Image Inpainting. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 7505–7514. https://doi.org/10.1109/CVPR42600.2020.00753

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 586–595. https://doi.org/10.1109/CVPR.2018.00068

Zhihua Zhong, Jingsen Zhu, Yuxin Dai, Chuankun Zheng, Guanlin Chen, Yuchi Huo, Hujun Bao, and Rui Wang. 2023. FuseSR: Super Resolution for Real-time Rendering through Efficient Multi-resolution Fusion. In *SIGGRAPH Asia 2023 Conference Papers, SA 2023, Sydney, NSW, Australia, December 12-15, 2023*, June Kim, Ming C. Lin, and Bernd Bickel (Eds.). ACM, 8:1–8:10. https://doi.org/10.1145/3610548.3618209

Tao Zhuang, Pengfei Shen, Beibei Wang, and Ligang Liu. 2021. Real-time Denoising Using BRDF Pre-integration Factorization. *Comput. Graph. Forum* 40, 7 (2021), 173–180. https://doi.org/10.1111/CGF.14411

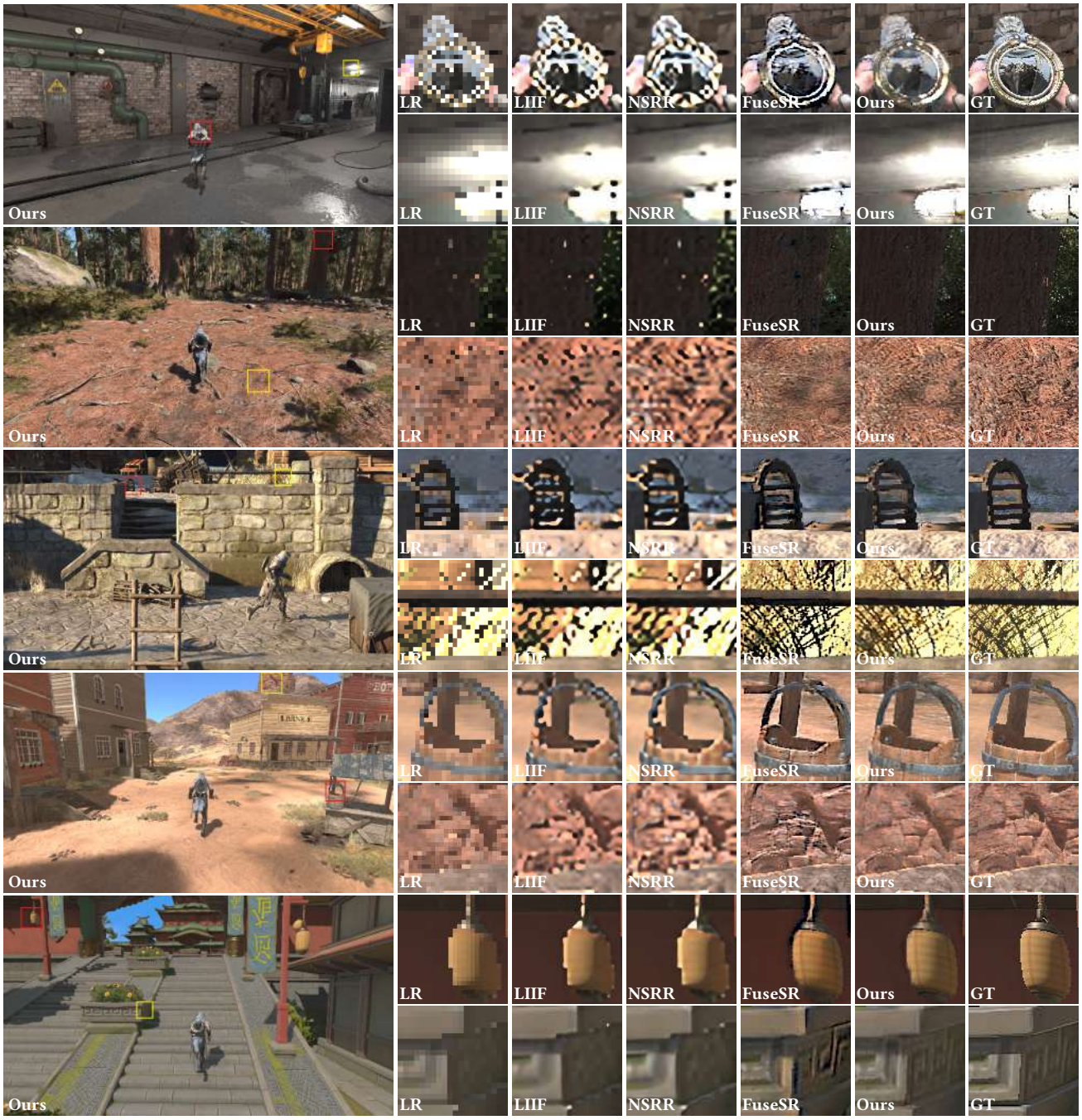

Figure 11: Visual comparisons of 4×4 super-resolution between our method and some baseline methods (LIIF [Chen et al. 2021b], NSRR [Xiao et al. 2020], and FuseSR [Zhong et al. 2023]) on the BK, RF, MD, WT and VE scenes. As seen, our method produces more sharp details and less artifacts than its competitors. Please refer to the supplementary video for animated sequences.

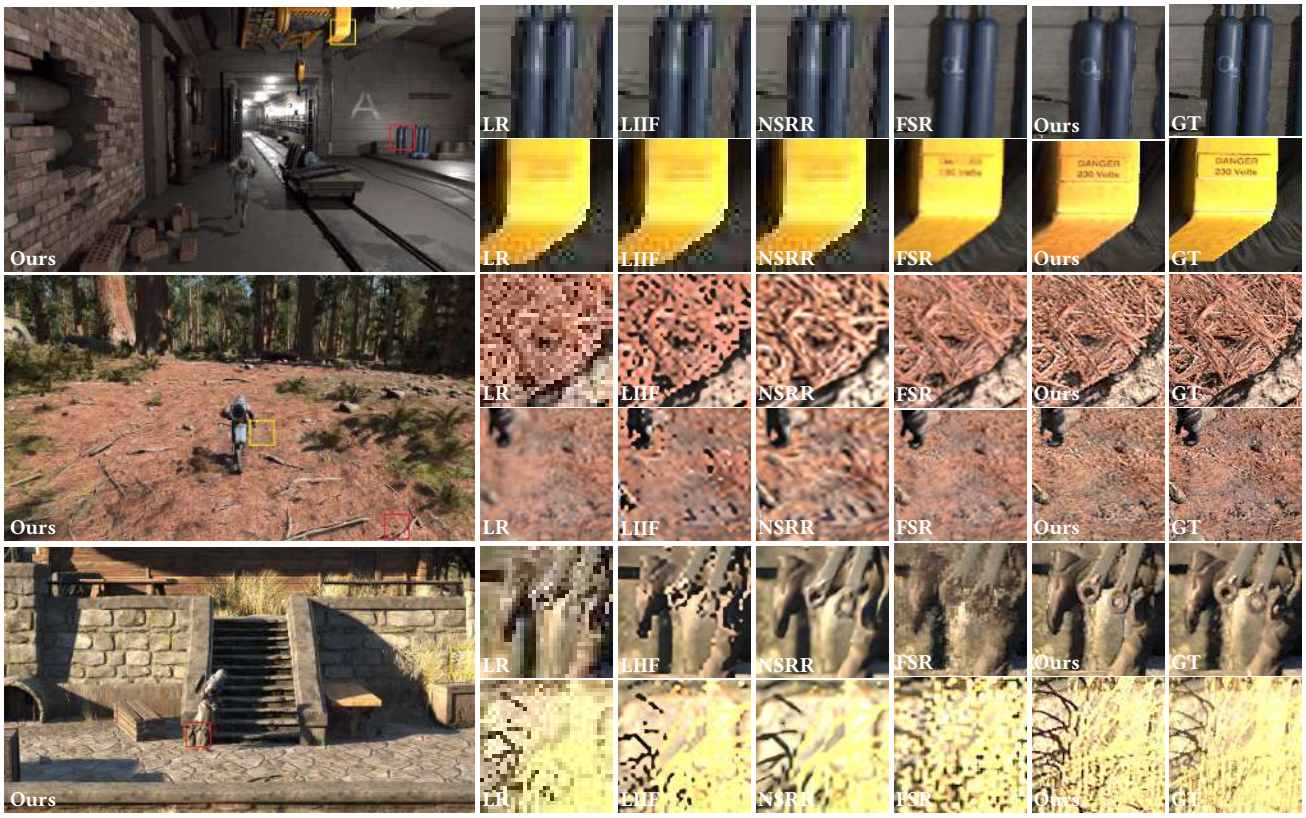

Figure 12: Visual comparisons of 3×3 super-resolution between our method and some baseline methods (LIIF[Chen et al. 2021b], NSRR [Xiao et al. 2020], and FSR [AMD 2023]) on the BK, RF, MD scenes .

Figure 13: Test the generalization ability of our model to 6×6 and 8×8 super-resolution. Note that both our model and LIIF are only trained on 2×2 to 4×4 cases.