Mob-FGSR: Frame Generation and Super Resolution for Mobile Real-Time Rendering

Sipeng Yang
12121024@zju.edu.cn
State Key Lab of CAD&CG, Zhejiang University
China

Qingchuan Zhu
22251012@zju.edu.cn
State Key Lab of CAD&CG, Zhejiang University
China

Junhao Zhuge
3200104439@zju.edu.cn
State Key Lab of CAD&CG, Zhejiang University
China

Qiang Qiu
qiang.qiu@oppo.com
OPPO Computing & Graphics Research Institute
USA

Chen Li
chenlixyz@gmail.com
OPPO Computing & Graphics Research Institute
USA

Yuzhong Yan
yuzhong.yan@oppo.com
OPPO Computing & Graphics Research Institute
USA

Huihui Xu
huihui.xu@oppo.com
OPPO Computing & Graphics Research Institute
USA

Ling-Qi Yan
lingqi@cs.ucsb.edu
University of California, Santa Barbara
USA

Xiaogang Jin*
jin@cad.zju.edu.cn
State Key Lab of CAD&CG, Zhejiang University
China

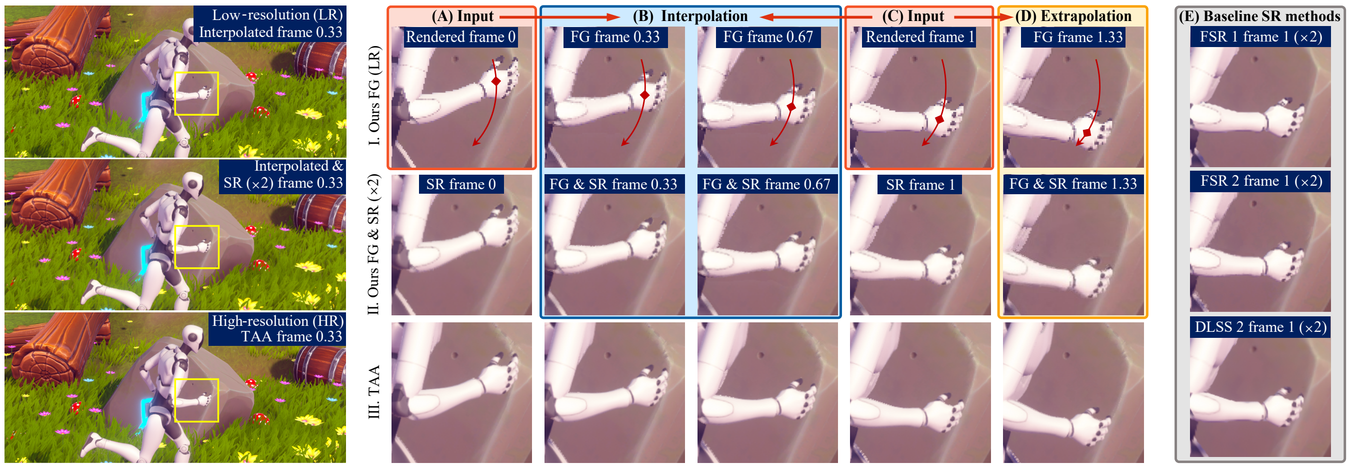

Figure 1: Frame generation (FG) and super resolution (SR) results of our method. Requiring only color, depth, and motion vectors from frames rendered at times 0 (A) and 1 (C), our approach efficiently produces interpolated (B) / extrapolated (D) frames and their SR counterparts at desired times. The left large images illustrate an interpolated frame (upper), its SR result (middle), and the high-resolution temporal anti-aliasing (TAA) reference (bottom). The right images show enlarged views of the yellow-framed regions of the interpolated frame (I.), their SR results (II.), and TAA references (III.), demonstrating our method's smooth motion estimation and high-quality SR results. (E) displays ×2 SR results of commercial applications FSR 1, FSR 2, and DLSS 2. Experiments show that our approach achieves results comparable to the best of these solutions. Notably, our models have short runtimes, processing FG at 720P in 2.2ms and FG & SR at 1080P in 2.3ms on a Snapdragon 8 Gen 3 processor.

ABSTRACT

Recent advances in supersampling for frame generation and super-resolution improve real-time rendering performance significantly. However, because these methods rely heavily on the most recent

[*]Corresponding author.

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0525-0/24/07…$15.00
https://doi.org/10.1145/3641519.3657424

features of high-end GPUs, they are impractical for mobile platforms, which are limited by lower GPU capabilities and a lack of dedicated optical flow estimation hardware. We propose *Mob-FGSR*, a novel lightweight supersampling framework tailored for mobile devices that integrates frame generation with super resolution to effectively improve real-time rendering performance. Our method introduces a splat-based motion vectors reconstruction method, which allows for accurate pixel-level motion estimation for both interpolation and extrapolation at desired times without the need for high-end GPUs or rendering data from generated frames. Subsequently, fast image generation models are designed to construct interpolated or extrapolated frames and improve resolution, providing users with a plethora of options. Our runtime models operate without the use of neural networks, ensuring their applicability to mobile devices. Extensive testing shows that our framework outperforms other lightweight solutions and rivals the performance of algorithms designed specifically for high-end GPUs. Our model's minimal runtime is confirmed by on-device testing, demonstrating its potential to benefit a wide range of mobile real-time rendering applications. More information and an Android demo can be found at: https://mob-fgsr.github.io/.

CCS CONCEPTS

• **Computing methodologies** → **Rendering**; **Image manipulation**.

KEYWORDS

Real-time rendering, supersampling, frame generation, super resolution

ACM Reference Format:
Sipeng Yang, Qingchuan Zhu, Junhao Zhuge, Qiang Qiu, Chen Li, Yuzhong Yan, Huihui Xu, Ling-Qi Yan, and Xiaogang Jin. 2024. Mob-FGSR: Frame Generation and Super Resolution for Mobile 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/10.1145/3641519.3657424

1 INTRODUCTION

Mobile platforms, such as smartphones, tablets, and head-mounted displays, are increasingly used for interactive digital entertainment and augmented reality, particularly mobile gaming. As a result, high-quality real-time rendering at high frame rates for mobile devices has become essential for providing users with a more fluid experience [Qualcomm 2023]. Modern smartphones have 2K resolution screens with refresh rates of 120Hz or 144Hz, providing unparalleled visual clarity and fluidity. However, limited GPU performance and power budgets impede efforts to improve mobile real-time rendering rates with high image quality, emphasizing the significance of advanced rendering acceleration techniques.

Several supersampling techniques have been developed to produce high-quality images with low rendering overhead. A class of methods exploit temporal frame redundancies [Briedis et al. 2023], employing frame reconstruction techniques that generate images directly via interpolation [Yang et al. 2011] or extrapolation [Guo et al. 2021]. Another strategy is to render at a low resolution (LR) and then use super-resolution (SR) models to generate high-resolution (HR) outputs [Edelsten et al. 2019]. Furthermore, recent advancements such as deep learning supersampling (DLSS) 3 [NVIDIA 2023], FidelityFX SR (FSR) 3 [AMD 2023], and ExtraSS [Wu et al. 2023a] integrate SR with frame generation, reducing rendering overhead significantly. These methods, however, typically necessitate high-end hardware for neural networks (NNs) execution or optical flow estimation, as well as specialized pipelines for extra G-buffers, rendering them unsuitable for mobile platforms.

To develop a framework for joint frame generation and SR on mobile platforms, we must address two key challenges: 1) Design fast and accurate motion reconstruction methods without the use of expensive optical flow from high-end hardware or motion vectors (MVs) of generated frames [Guo et al. 2021] from deferred rendering. Existing lightweight motion reconstruction approaches, such as 3DWarp [Mark et al. 1997] and bidirectional scene reprojection (BSR) [Yang et al. 2011], while fast and hardware-independent, frequently produce inaccurate estimates for disocclusions or thin objects, resulting in visual artifacts. 2) Develop efficient models for generated and SR frame reconstruction to achieve the desired high frame rates. Although NNs have become the de facto standard for achieving real-time SR in recent years, they are difficult to implement in off-the-shelf mobile devices. Even the mobile-optimized SR model MNSS [Yang et al. 2023], which deploys NNs on mobile SoC AI units, requires an additional 11ms of runtime for 720P image generation, making it unsuitable for today's mobile phones.

To address the aforementioned challenges, we present a fast supersampling framework for mobile platforms that integrates frame generation and SR to improve real-time rendering performance. Our framework is based on three critical components: (1) accurate MVs reconstruction based on splats, (2) high-quality generated frame reconstruction, and (3) fast resolution upscaling. At the heart of our approach is the fast and accurate MVs reconstruction for generated frames, which is based solely on data from rendered frames. Specifically, we estimate pixel motion in generated frames using MVs from rendered frames under the assumption of quadratic motion between adjacent frames. After that, the splatting method with atomic operations [Khronos 2022] is used to construct pixel motion to MVs at desired times. Then, using the generated MVs, we create lightweight supersampling models to generate high-quality interpolated and extrapolated frames as well as their SR counterparts. Furthermore, rigorous experiments are carried out to optimize the framework design for mobile platforms and calibrate key parameters to ensure optimal performance. Importantly, our runtime models are devoid of NNs, making them ideal for off-the-shelf mobile devices.

Our framework supports four modes: interpolation, extrapolation, and SR-enhanced variants of both. Our approach, relying solely on color images, depth maps, and MVs of rendered frames, not only allows for fast frame generation at the desired time, but also ensures producing high-fidelity, resolution-enhanced, and anti-aliased outputs. The proposed method stands out due to its short runtime, which perfectly aligns with the demands for mobile real-time rendering. In summary, our method makes the following contributions:

- A splat-based MVs reconstruction method that estimates accurate pixel-level motion and constructs MVs for both interpolated and extrapolated frames at desired times.

- A lightweight supersampling framework tailored for real-time rendering on mobile devices for frame generation and SR.
- An optimized module design that balances performance and quality, as well as data-driven parameter settings for mobile supersampling.

2 RELATED WORK

2.1 Antialiasing and Supersampling

Aliasing artifacts, manifested as jagged contours and moiré patterns, are common in rendered images as a result of insufficient sampling rates [Akenine-Mo et al. 2018]. A direct solution is to increase the samples per pixel, as seen in SSAA [Cook 1986], but this significantly increases the computational demand of rendering. Akeley [1993] introduced MSAA to improve rendering performance by reducing the number of shading operations to minimize rendering overhead. Additionally, morphological antialiasing techniques, such as MLAA [Reshetov 2009] and SMAA [Jimenez et al. 2012], are proposed to reduce jagged edges by applying filtering to post-rendered images, balancing efficiency and image quality.

In the past decade, the advent of temporal antialiasing (TAA) [Yang et al. 2020] has signified a major shift in antialiasing from spatial to temporal supersampling. Utilizing consecutive frames, TAA and its variants efficiently distribute the shading workload over time, achieving effective antialiasing with a marginally additional cost. Expanding on this concept, recent research has explored utilizing the abundant temporal samples accumulated by TAA to directly reconstruct higher resolution frames. For instance, temporal SR (TSR) [Epic 2022] and FSR 2 [AMD 2022] employ heuristic methods for antialiased HR frame reconstruction. Concurrently, solutions like DLSS 1 [Edelsten et al. 2019], DLSS 2 [Liu 2020], reduced-precision network [Thomas et al. 2020], neural supersampling for real-time rendering (NSR) [Xiao et al. 2020], MNSS [Yang et al. 2023], and FuseSR [Zhong et al. 2023] utilize NNs for enhanced image reconstruction, and some of these advanced techniques are becoming standard in PC video gaming. Our work is also based on temporal supersampling, but it focuses on the practical application of supersampling for mobile platforms.

2.2 Frame Generation for Rendering

Another method for improving rendering performance is to generate interpolated or extrapolated frames directly. To produce the generated frame, the core principle of frame generation methods involves estimating pixel motion across consecutively rendered frames. Therefore, accurately modeling pixel's motion emerges as a critical aspect of successful frame generation. An early attempt, 3DWarp [Mark et al. 1997], proposes using the temporal positional changes of pixels relative to camera to determine pixel motion. However, this approach frequently encounters difficulties in disoccluded areas, resulting in suboptimal projections. Tracking inter-frame pixel motions becomes easier with the inclusion of MVs in rendering engines. BSR [Yang et al. 2011], iterative image warping [Bowles et al. 2012], and iterative depth warping [Lee et al. 2018] expand on this by introducing iterative-based methods for predicting motion in the generated frame based on rendered MVs. Nonetheless, these method has limitations with intricate or rapidly moving objects, frequently producing inaccurate predictions.

Table 1: Key capabilities of frame generation (in cornsilk), SR (in light pink), and joint solutions (in Alice blue) relevant to our research. Our method supports interpolation, extrapolation, and SR. It also supports forward rendering (as opposed to deferred rendering only), aligning with mainstream mobile rendering pipelines. Our lightweight method is also designed to be highly efficient for mobile devices.

Methods	Interpolation	Extrapolation	Super Resolution	Forward Rendering	Mobile Friendly
3DWarp [1997]	○				○
BSR [2011]	○			○	○
ExtraNet [2021]		○			
TSR [2022]			○	○	○
NSR [2020]			○	○	
DLSS 2 [2020]			○	○	
FSR 2 [2022]			○	○	○
MNSS [2023]			○	○	○
DLSS 3 [2023]	○		○	○	
FSR 3 [2023]	○		○	○	
ExtraSS [2023a]		○	○		
Ours	○	○	○	○	○

Beyond the rendered MVs, recent studies explore optical flow estimation for frame generation. Interpolation methods like learnable MVs [Wu et al. 2023b], NFI [Briedis et al. 2021] and KBI [Briedis et al. 2023] leverage estimated optical flow or MVs to generate unshaded frames. DLSS 3 [NVIDIA 2023], a notable commercial application, also follows a similar approach, but its specific details remain undisclosed. Extrapolation methods like future frame synthesis [Li et al. 2022], ExtraNet [Guo et al. 2021], and ExtraSS [Wu et al. 2023a] use NNs to estimate optical flow or lighting variations for generated frames. Although these methods, which utilize NNs and costly optical flow estimation, deliver high-quality visuals, their computational intensity presents challenges for resource-limited mobile platforms. Furthermore, while rendering MVs of the generated frame in deferred shading is trivial on high-end hardware, it imposes a significant burden on mobile platforms, making efficient frame generation on mobile devices more difficult. Refer to Tab. 1 for a clear organization of the relevant work's attributes of interest.

There are also some fast motion estimation methods that are independent of NNs and rendered MVs, such as using the fast patch-match algorithm for mapping calculations of adjacent frames [Barnes et al. 2010; Hu et al. 2016]; and utilizing iterative schemes and image pyramids for searching pixel correspondences across frames [Hanika et al. 2021]. However, the estimated motion from these methods often exhibits lower accuracy and resolution, which would diminish the quality of the generated frames. Moreover, despite their lightweight design, attaining high frame rates on mobile devices remains a significant challenge using these approaches.

3 METHOD

3.1 Definitions

Our goal is to develop an integrated model for frame generation and SR. To ensure that the model is applicable to off-the-shelf mobile devices, our chosen inputs consist of typical forward rendering outputs: color image I, depth map D, and MVs M. We use rendering

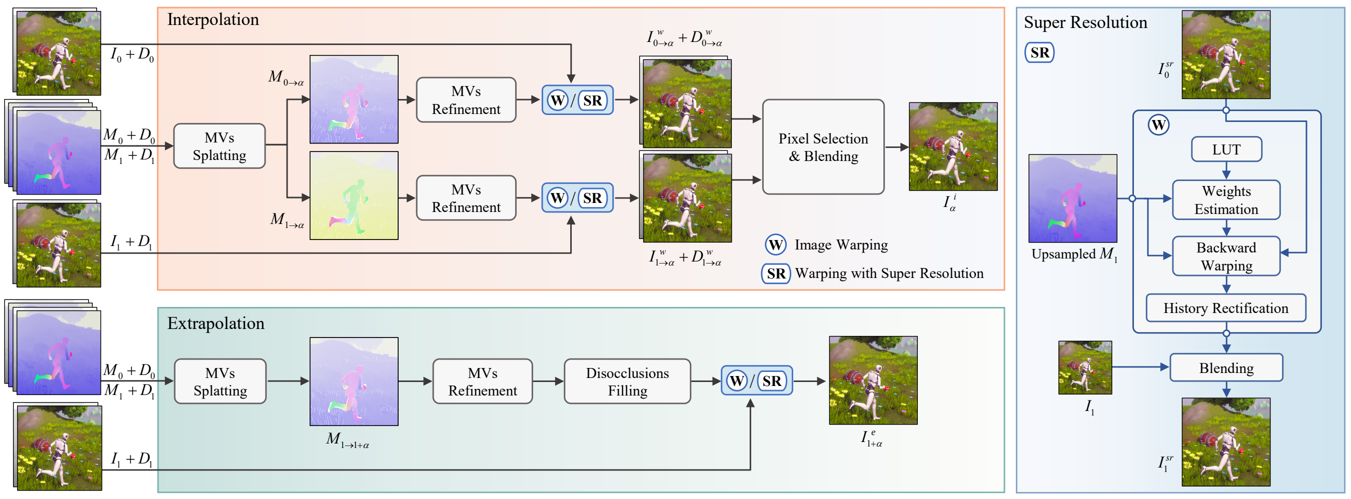

Figure 2: Framework overview. For interpolation, MVs reconstruction of the B-frame I^i_α is performed using MVs and depth from I-frames 0 and 1, and color and depth information from these I-frames are then utilized to construct the final output I^i_α. In extrapolation, MVs and depth from I-frames 0 and 1 facilitate motion estimation, with color and depth from I-frame 1 employed to generate the extrapolated P-frame $I^e_{1+\alpha}$. For their SR versions, the image warping module is replaced by warping with SR. In this process, resampling weights are retrieved from a lookup table (LUT) based on MVs sampling positions, followed by backward warping. Note that warping with SR only involves using the SR image from the previous I-frame.

data from frames at times 0 and 1 as input for both interpolation and extrapolation, which are denoted as $F_0 = \{I_0, D_0, M_0\}$ and $F_1 = \{I_1, D_1, M_1\}$, respectively. Our interpolation and SR model, denoted as *Ours-ISR*, produces the output image I^{isr}_α. The superscript 'i' denotes interpolation, and 'sr' signifies SR. Similarly, our extrapolation and SR model (*Ours-ESR*) produces the image $I^{esr}_{1+\alpha}$, and superscript 'e' denotes extrapolation. Our interpolation-only model (*Ours-I*) and extrapolation-only model (*Ours-E*) generate images I^i_α and $I^e_{1+\alpha}$, respectively. The subscript α marks the temporal position of the generated frames, with $\alpha \in (0,1)$ for interpolation and $1+\alpha$ for extrapolation (taking the frame at time 1 as the current frame). For a standardized representation, we adopt terminology from video encoding [Richardson 2004]: rendered frames are referred to as *I-frames* (Intra-coded), interpolated ones as *B-frames* (Bidirectional predicted), and extrapolated ones as *P-frames* (Predicted).

3.2 Method Overview

Fig. 2 shows the architecture of our proposed framework, which primarily consists of two main parts: frame generation (including interpolation and extrapolation) and SR. For the *interpolation* model, data from two consecutive I-frames 0 and 1 are used to generate a B-frame. Initially, MVs and depth from I-frames 0 and 1 are utilized for motion splatting to obtain the bidirectional MVs $M_{0\to\alpha}$ and $M_{1\to\alpha}$, which represent the pixels' motion from the two I-frames to a B-frame I^i_α. Since the derived MVs $M_{0\to\alpha}$ and $M_{1\to\alpha}$ frequently present grid-like gaps, we introduce a refinement module to fix these issues. After that, the color and depth from the two I-frames are aligned to the B-frame by image backward warping. Finally, a pixel selection & blending module blends the aligned two frames to produce the interpolated result I^i_α. For the *extrapolation* model, we initially predict single-direction MVs $M_{1\to 1+\alpha}$ for the P-frame $I^e_{1+\alpha}$. To avoid ghosting and flickering issues, a disocclusion filling module is then employed to address the absence of MVs in disoccluded regions. Finally, we execute backward warping exclusively on I-frame 1 to produce the extrapolated frame $I^e_{1+\alpha}$.

We design an optional SR module to improve image resolution by using accumulated temporal samples. To construct the SR frame I^{sr}_1, we warp the history SR frame I^{sr}_0 to align with the current frame (Fig. 2, right part), rectify invalid pixels in the warped history frame, and blend it with the LR frame I_1 to obtain our SR result. This SR module can be seamlessly integrated into the frame generation process as a replacement for the standard image warping module.

3.3 MVs Reconstruction

In the absence of rendered MVs from deferred rendering or optical flow estimates, our method begins by estimating MVs for B/P-frames using available I-frame data. We introduce a fast splat-based motion estimation method that leverages depth and MVs from rendered I-frames to construct MVs for generated frames. By analyzing two consecutive frames, our method is able to estimate precise pixel motions that depict linear and nonlinear trajectories with uniform acceleration (quadratic motion). As shown in Fig. 3, considering a thrown green ball and its trajectory, we mark its positions at I-frames -1, 0, and 1 as p_{-1}, p_0, and p_1, respectively. I-frame 1 is treated as the current frame. Utilizing MVs $\boldsymbol{m}_0$ and $\boldsymbol{m}_1$ in image space as input, we derive positions p_α and $p_{1+\alpha}$ for desired interpolated time α or extrapolated time $1+\alpha$, following the formulas in Fig. 3. The motions of pixels in generated frames are then obtained, with vectors $\boldsymbol{m}_{0\to\alpha}$ and $\boldsymbol{m}_{1\to\alpha}$ for interpolation, and $\boldsymbol{m}_{1\to 1+\alpha}$ for extrapolation. Detailed formula derivations are provided in the supplementary materials.

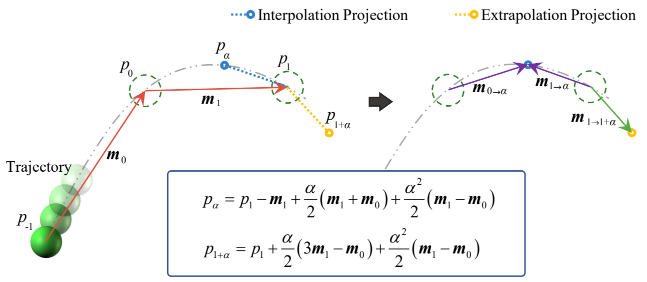
Interpolation Projection
Extrapolation Projection
p_α
p_0
p_1
m_1
$p_{1+\alpha}$
$m_{0\to\alpha}$
$m_{1\to\alpha}$
$m_{1\to1+\alpha}$
Trajectory
m_0
p_{-1}
$p_\alpha = p_1 - m_1 + \frac{\alpha}{2}(m_1 + m_0) + \frac{\alpha^2}{2}(m_1 - m_0)$
$p_{1+\alpha} = p_1 + \frac{\alpha}{2}(3m_1 - m_0) + \frac{\alpha^2}{2}(m_1 - m_0)$

Figure 3: Motion estimation process. A green ball is thrown, tracing a trajectory marked at p_{-1}, p_0, and p_1 in temporal frames. Taking p_1 in frame 1 as the reference, we estimate the ball's positions p_α in the interpolated frame and $p_{1+\alpha}$ in the extrapolated frame. By modeling the motion as uniformly accelerated, p_α and $p_{1+\alpha}$ are deduced from prescribed formulas. The resulting motions, $\boldsymbol{m}_{0\to\alpha}$ and $\boldsymbol{m}_{1\to\alpha}$, and $\boldsymbol{m}_{1\to1+\alpha}$ are then accurately splatted onto their respective positions.

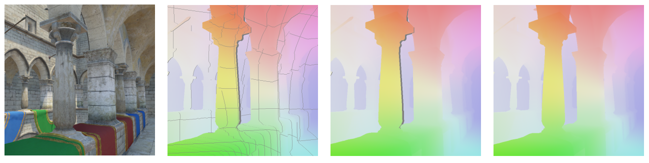

Figure 4: MVs refinement. From left to right: the color image, splatted MVs prior to refinement, MVs post-refinement, and the ground truth MVs for reference.

The MVs construction process continues by embedding the computed vectors into their respective pixels. This process involves assigning $\boldsymbol{m}_{0\to\alpha}$ at position p_α, a process known as 'splatting' [Shade et al. 1998]. Given that the estimated position p_α usually does not align exactly with a pixel's center, we utilize the nearest neighbor method to identify the closest pixel for writing vectors. This splatting is applied to each pixel in I-frame 1 to generate the projected MVs $M_{0\to\alpha}$, $M_{1\to\alpha}$, and $M_{1\to1+\alpha}$. Note that conflicts can occur during the splatting process, particularly at object edges where multiple vectors are projected onto a single pixel. To address this, we implement depth-aware MVs splatting along with atomic operations [Khronos 2022], ensuring that the MVs with minimal depth (i.e., foreground pixels) are written when conflicting. Furthermore, due to the nearest neighbor approach used during splatting, the projected MVs frequently have grid-like gaps, as shown in Fig. 4(b). Hence, we use a thin-object detection method inspired by FSR 2 [AMD 2022] to identify gaps and then fill them with a mean filter. This method identifies thin objects by examining each pixel's 3×3 area to detect the absence of any 2×2 pixel block with similar features. Our tests have shown that it is both efficient and robust. The refinement step yields the repaired MVs shown in Fig. 4 (c).

In general, the quadratic motion assumption is particularly apt for our task due to its low computational overhead and satisfactory visual outcomes. Xu et al. [2019] also propose using quadratic motion estimation in video interpolation tasks to enhance visual

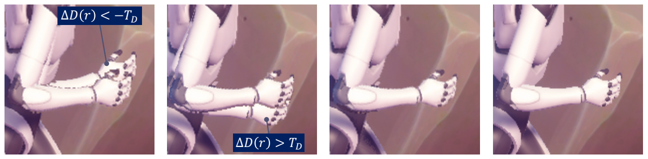

Figure 5: B-frame reconstruction. From left to right are the warped two I-frames $I^w_{0\to\alpha}$ and $I^w_{1\to\alpha}$, our reconstructed B-frame I_α, and the reference image.

effects. However, their method estimates quadratic "flow" at time 0, neglecting the quadratic motion during the splatting phase (from time 0 to time α); in contrast, our method directly calculates the quadratic "position" at time α, which is theoretically more accurate. Additionally, the Gaussian-based splatting used by Xu et al. [2019] is computationally intensive and ineffective at detecting disoccluded regions essential for subsequent image reconstruction. Therefore, we prefer using our approach to reconstruct MVs for generated frames due to its superior efficiency and accuracy.

3.4 Frame Generation

Interpolation. After obtaining the derived MVs, we can use them and the rendered I-frames to construct B/P-frames. For interpolation, two adjacent I-frames are warped using MVs $M_{0\to\alpha}$ and $M_{1\to\alpha}$, respectively. The warped two frames, $I^w_{0\to\alpha}$ and $I^w_{1\to\alpha}$, are then blended to create the B-frame I^i_α. During this blending process, we face two major challenges. First, slight inaccuracies in the estimated bidirectional MVs can lead to blurring when directly averaging the two warped frames. To mitigate this, we preferentially sample color from the temporally closer frame when the color and depth of the two warped frames are similar. Second, it is crucial to address artifacts that emerge as disocclusions and regions with shading changes in the warped two frames. For disocclusions, we select the appropriate pixels from both frames to avoid ghosting artifacts (see Fig. 5). For shading changes, we use a simple linear interpolation (*lerp*) of the two frames to provide a smooth transition. The above strategy is expressed as follows:

$$I^i_\alpha(r) = \begin{cases} I^w_{0\to\alpha}(r), & \Delta D(r) > T_D, \\ I^w_{1\to\alpha}(r), & \Delta D(r) < -T_D, \\ I^w_{0\to\alpha}(r), & \alpha \leqslant 0.5 \;\&\; |\Delta D(r)| \leqslant T_D \;\&\; |\Delta B(r)| \leqslant T_B; \\ I^w_{1\to\alpha}(r), & \alpha > 0.5 \;\&\; |\Delta D(r)| \leqslant T_D \;\&\; |\Delta B(r)| \leqslant T_B; \\ \mathrm{lerp}(I^w_{0\to\alpha}(r), I^w_{1\to\alpha}(r), \alpha), & |\Delta D(r)| \leqslant T_D \;\&\; |\Delta B(r)| > T_B; \end{cases} \tag{1}$$

where r denotes regions that meet the specified conditions, ΔD represents the depth difference between the warped depth maps $D^w_{0\to\alpha}$ and $D^w_{1\to\alpha}$, ΔB signifies the brightness difference between images $I^w_{0\to\alpha}$ and $I^w_{1\to\alpha}$, and T_D and T_B are thresholds. Typically, disocclusions are indicated by large depth differences, whereas shading changes are indicated by significant brightness differences despite similar depths. Thresholds T_D and T_B are calibrated using a data-driven approach, as described in Sec. 4.

Extrapolation. The absence of subsequent frame references creates difficulties in addressing disocclusions in the construction of the extrapolation frame $I^e_{1+\alpha}$. To address this issue, several solutions have been proposed, such as background filling [Andreev 2010; Schollmeyer et al. 2017], occlusion MVs [Zeng et al. 2021], and G-buffer guided warping [Wu et al. 2023a]. However, all these methods will incur considerable computational overhead, impeding the realization of high frame rates on mobile platforms. Therefore, we present a simplified disocclusion filling approach that directly uses the MVs of I-frame 1 for filling disocclusions in the MVs of extrapolated frames. This method strikes an effective balance between runtime efficiency and image quality (see performance comparison in Sec. 4). It should be noted that considerations of *shading changes* are omitted in our extrapolation approach because they typically necessitate a large amount of computational power. This compromise may result in lower frame rates in areas with moving shadows [Guo et al. 2021] (see our supplementary video).

3.5 SR Integration

Following the generation of interpolated or extrapolated frames, we introduce a standalone SR module to enhance image resolution. As depicted in Fig. 2, this lightweight SR module can replace the image warping module to increase spatial resolution. This module adopts a common framework, reusing samples from history frames to construct the HR current image I^{sr}_1.

To begin, we use backward warping to align the history frame I^{sr}_0 with the current one. Bilinear or bicubic interpolations are frequently used for image warping. However, during multiple resampling of the frame reusing process, bilinear interpolation frequently produces blurred results [Yang et al. 2020], and bicubic interpolation is computationally intensive for mobile devices. As a result, we introduce a lookup table (LUT)-based method for efficient image warping on mobile platforms, which achieves the same quality as bicubic interpolation while incurring less computational overhead. The LUT stores resampling weights, which are based on sampling positions. To create this LUT, we pre-train a neural network to predict pixel weights during the iterative resampling process, aiming for the best warping results. These weights are then quantized and saved in the LUT for quick access during inference on mobile devices. Section 4 provides additional experimental details and comparisons to bilinear and bicubic interpolation methods.

After that, we rectify invalid pixels in areas of disocclusion and shading changes in the warped history frame before blending it with the current I-frame I_1. As the detection process is similar to that described in Eq. 1, we use the thresholds T_D and T_B directly to detect invalid pixels. Pixels within disocclusion regions are directly rejected. Similar to the TAA method, history colors are clamped against the AABB of the current frame to address shading changes. Finally, the LR I-frame I_1 is blended with the warped and rectified history frame I^{sr}_0 to create the SR result I^{sr}_1 by:

$$q = a \cdot \frac{\sum_{s \in \Omega_p} (1 - b \cdot d_s) \cdot s}{n} + (1 - a) \cdot q^w, \tag{2}$$

where $q \in I^{sr}_1$ is the constructed pixel, $q^w \in I^{sr,w}_0$ is the corresponding warped history pixel, $s \in I_1$ denotes pixel samples near q within range Ω_p (a radius of one LR pixel width), n is the number of samples s in Ω_p, and d_s is the distance from s to q. The tunable parameters a and b are discussed in Sec. 4.

We use low-latency solutions to perform SR for generated B/P frames. In the interpolation and SR method, SR I-frame 0 (I^{sr}_0) and LR I-frame 1 (I_1) are used to generate frame I^{sr}_α. In the extrapolation and SR method, the history SR I-frame I^{sr}_1 is directly employed to construct the HR extrapolated frame $I^{esr}_{1+\alpha}$. Except for the required upsampling of corresponding depth maps and MVs, the remaining operations are carried out in a manner similar to our interpolation-only and extrapolation-only methods.

4 DATA-DRIVEN OPTIMIZATION

This section delves into the parameter tuning in our framework. We set up six game scenes in the Unity Engine [Unity 2023] to create an evaluation environment. Images rendered with 9× SSAA serve as the ground truth for SR. The tests are carried out on a Snapdragon 8 Gen 3 smartphone with a 1080P resolution.

Learning LUT. In Sec. 3.5, we introduced learned resampling weights for repeated image warping on mobile platforms. This approach employs a data-driven strategy to learn the most effective filters for repeated sampling. Specifically, we train a multilayer perceptron (MLP) to learn pixel weights based on offsets d_x and d_y from a sample s to its nearest upper-left pixel, as shown in Fig. 6. After training, d_x and d_y are quantized into 32 discrete levels, and the sampling weights derived from the MLP are stored in a 32×32×16 LUT. During inference, this LUT provides quick access to the weights of 16 pixels based on their sampling positions. The LUT does not require retraining for each scene but needs to be trained for different upscaling factors. More details about the training process can be found in the supplementary material. Our experimental results show that the LUT-based method achieves a high mean SSIM of 0.933 (compared to bilinear's 0.907 and bicubic's 0.931), while maintaining a short runtime of 0.905ms (faster than bicubic's 1.145ms). More importantly, the LUT-based method can be introduced more information, such as color gradients, in the lookup process, potentially improving results.

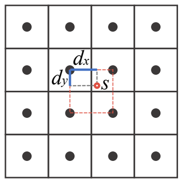

Figure 6: Sampling.

Parameters calibration. In Sec. 3.4, thresholds T_D and T_B are utilized in Eq. 1 to validate history pixels for the reuse of previous SR images. We adopt a data-driven approach, treating these thresholds as learnable parameters and optimizing them using a deep learning framework. Our tests across various scenes indicate that the model performs optimally at $T_D = 0.0028$ and $T_B = 0.13$. Additionally, the method for calculating pixel blending weights is detailed in Eq. 2. In our testing environment, we calibrate the coefficients $a = 0.34$ and $b = 0.81$ to ensure optimal blending results.

5 RESULTS AND COMPARISONS

This section evaluates the performance of our method compared to existing solutions. To prevent test data leakage, we create four new Unity scenes for our evaluation, distinct from the six previously used. PSNR and SSIM are employed as metrics for quantitative assessment. It is worth noting that some supersampling techniques are unique to Unreal Engine 5 (UE) [Epic Games 2023] and its deferred rendering process. Therefore, we also develop two UE scenes for a direct and fair comparison with these specific methods.

5.1 Comparisons on Unity scenes

The performance of our method is first assessed across four Unity scenes: *Street View* (SV), *Meadows* (ME), *Hilly Area* (HA), and *Dragon Park* (DP). Evaluations are divided into three parts: frame generation, SR, and combined operations. Unless otherwise specified, experiments are set to "single frame generation" and "×2 resolution upscaling" for frame generation and SR, respectively.

Frame Generation. We compare our models, Ours-I and Ours-E, with existing frame generation methods, including 3DWarp [Mark et al. 1997], BSR [Yang et al. 2011] and AFME[1] [Holmes and Wicks 2020]. Note that all these methods are potentially for real-time execution on mobile platforms. The quantitative evaluation results, as reported in Tab. 2 (rows in cornsilk), clearly demonstrate that our frame generation models significantly outperform other lightweight methods, achieving a minimum improvement of 0.049 in mean SSIM and 4.22dB in mean PSNR. Visual comparisons are illustrated in Fig. 7. It is observable that AFME, which relies on LR optical flow estimation, often leads to noticeable distortion artifacts in the output. In cases of fast-moving thin objects or disoccluded regions, 3DWarp and BSR yield incorrect motion estimations, resulting in unsatisfactory results. In contrast, our method provides accurate pixel-level motion estimation, achieving markedly better results.

SR. For the comparison with real-time SR methods, we evaluate the I-frames SR results of our method (Ours-SR) against the deep learning-based solutions NSR [Xiao et al. 2020] and MNSS [Yang et al. 2023]. The quantitative and qualitative results, as shown in Tab. 2 (rows in light pink) and Fig. 7, indicate that our results are marginally lower but competitive with these deep learning methods.

Joint Solution. In Tab. 2 (rows in Alice blue), we present evaluations of our joint frame generation and SR models. Compared to Ours-SR, Ours-ISR and Ours-ESR show slight performance reductions. This is mainly due to the misalignment of generated frames with reference images, arising from minor variances between our estimated motion and the ground truth. In fact, the image quality of Ours-ISR and Ours-ESR is closely comparable to that of Ours-SR, as shown in the lower part of Fig. 7.

5.2 Comparisons on UE scenes

We also extend our comparisons to two new UE scenes with deferred shading: *Bunker* (BK) and *Western Town* (WT). We collect essential data for ExtraNet [Guo et al. 2021] to facilitate its model training. Our evaluations include comparisons with TSR [Epic 2022], FSR 2 [AMD 2022], and DLSS 2 [Liu 2020]. Tab. 3 shows the quantitative results. In terms of frame generation, Ours-I and Ours-E lag behind ExtraNet. This gap is primarily caused by ExtraNet's direct acquisition of MVs and the generated frame's base color, which ensures perfect alignment between the generated frame and the reference image. In contrast, our estimated MVs have minor deviations compared to ground truth, reasonably resulting in lower PSNR and SSIM performance. In metrics, the Ours-SR method slightly outperforms TSR for SR tasks. The visual comparisons in Fig. 8 support these findings. Predicting optical flow using deep learning models (or implicit alternatives derived from history frames, as demonstrated by ExtraNet) does improve shading variation prediction. It is noteworthy that TSR, FSR2, and our method achieve comparable SR results; however, these non-NN methods tend to produce rough edges at fast-moving and newly emerged objects. In contrast, by employing deep NNs for image reconstruction, DLSS 2 consistently delivers superior SR results with smoother edges. The superiority of DLSS 2, as well as NSR and MNSS methods, stems from the trained NNs' ability to suppress noise at low sampling rates and to predict high-quality edges and textures. This capability demonstrates the benefits of deep learning and identifies potential areas for future research on non-NN solutions.

[1]Qualcomm's Adreno Frame Motion Engine (AFME) is engineered for Adreno GPUs to increase video frame rates. This is achieved by analyzing motion across consecutive frames and synthesizing additional frames, thereby enhancing visual fluidity.

Table 2: Comparisons with existing methods on Unity scenes. Our approach is compared separately for frame generation (cornsilk), SR (light pink), and their joint solutions (Alice blue). The evaluation metrics used are PSNR and SSIM.

Methods	PSNR↑				SSIM↑			
	SV	HA	ME	DP	SV	HA	ME	DP
3DWarp	22.70	26.62	29.21	29.19	0.792	0.912	0.906	0.893
BSR	21.91	26.07	27.14	27.90	0.771	0.907	0.897	0.875
AFME	20.87	24.81	25.79	26.98	0.732	0.875	0.832	0.814
Ours-E	26.60	30.85	31.53	30.65	0.854	0.934	0.945	0.908
Ours-I	**27.57**	**31.78**	**32.47**	**32.77**	**0.878**	**0.948**	**0.952**	**0.922**
NSR	**29.07**	**33.56**	**34.22**	30.41	**0.896**	0.922	**0.941**	0.917
MNSS	28.95	32.90	32.17	**30.87**	0.890	**0.929**	0.934	**0.923**
Ours-SR	28.42	31.73	32.36	30.62	0.887	0.910	0.932	0.915
Ours-ESR	25.47	29.85	30.61	30.03	0.843	0.898	0.916	0.904
Ours-ISR	26.94	31.03	31.89	30.49	0.861	0.902	0.927	0.912

Table 3: Comparisons with existing methods on UE scenes.

	Scenes	ExtraNet	Ours-E	Ours-I	TSR	Ours-SR	Ours-ESR	Ours-ISR
PSNR	BK	**31.19**	28.19	29.47	29.07	**30.85**	27.91	28.57
	WT	**28.97**	25.63	26.22	28.54	**28.89**	25.52	26.80
SSIM	BK	**0.950**	0.902	0.914	0.910	**0.923**	0.881	0.894
	WT	**0.912**	0.887	0.891	0.874	**0.885**	0.849	0.863

5.3 Performance

We perform a thorough analysis of our models' runtime on devices equipped with modern mobile processors, specifically the Qualcomm Snapdragon 8 Gen 3. As shown in Tab. 4, our models are extremely efficient. Ours-I and Ours-E achieve frame generation runtimes of 2.2 ms and 1.9 ms, respectively, at 720P resolution. Ours-SR, Ours-ISR, and Ours-ESR achieve runtimes of 1.7 ms, 2.3 ms, and 1.8 ms, respectively, for frame generation and SR from 540P to 1080P. Compared to other methods listed at the bottom of Tab. 4, 3DWarp is slower due to additional rasterization, while BSR and AFME, though faster, yield less satisfactory results (see Fig. 7). Despite MNSS's use of model quantization and hardware acceleration, Ours-SR remains nearly seven times faster. Overall, our method provides high-quality frame generation and SR in a very short time, making it ideally suited for improving current mobile real-time rendering applications.

Table 4: Runtime performance analysis. We report the inference times of our models on a modern mobile SoC (Snapdragon 8 Gen3) and compare them with alternative solutions.

	Frame generation (720P)		Frame generation & SR (1080P)		
Modules	Ours-I	Ours-E	Ours-SR	Ours-ISR	Ours-ESR
MVs Splatting	0.75	0.67	-	0.43	0.38
Refinement	0.57	0.49	-	0.32	0.29
Disocc. Filling	-	0.20	-	-	0.18
Warping	0.77	0.56	1.24	1.35	0.90
Blending	0.12	-	0.42	0.24	-
Total	2.21ms	1.92ms	1.66ms	2.34ms	1.75ms

	Frame generation (720P)			SR (1080P)
Alternative Methods	3DWarp	BSR	AFME	MNSS
Runtime	24.16ms	3.11ms	<1ms	12.84ms

6 CONCLUSIONS

We presented a lightweight supersampling framework that boosts mobile real-time rendering. By combining frame generation and SR, we significantly reduce rendering overhead, increase frame rates, and maintain high visual quality. Our framework introduces an MVs reconstruction method based on splats, which ensures precise motion estimation for generated frames at desired times. Furthermore, fast frame generation and SR models are intended to generate high-fidelity interpolated and extrapolated frames, as well as their SR variants, providing users with a variety of options. Finally, after extensive research, we identified appropriate framework designs for mobile platforms, ensuring optimal model performance. Comprehensive tests show that our framework outperforms other lightweight methods and approaches the performance of algorithms designed specifically for high-end GPUs. Our models are highly efficient, and they are useful for a wide range of mobile devices such as smartphones, handheld consoles, and head-mounted displays.

Our method has limitations. 1) We ignore shading changes during frame generation due to computational constraints, resulting in low frame rate effects on shadows, reflections, and transparent objects (see Fig. 9). We can partly mitigate this issue by using soft shadows and avoiding large reflections and transparent materials. On the other hand, directly using deep NNs to predict shading changes on mobile hardware remains a difficult task. 2) Our MVs reconstruction method supports extrapolation to reduce frame generation latency but may introduce ghosting artifacts in disoccluded areas. We recommend an effective two-frame generation strategy: the first frame utilizes extrapolation for low latency, while the second frame uses interpolation to reduce artifacts. This approach maximizes the benefits of extrapolation and minimizes artifacts (see the demo video for improvements). Future research could address this issue by exploring lightweight disocclusion filling methods akin to G-buffer guided warping [Wu et al. 2023a].

ACKNOWLEDGMENTS

Xiaogang Jin was supported by Key R&D Program of Zhejiang (No. 2024C01069), the National Natural Science Foundation of China (Grant No. 62036010), and the FDCT under Grant 0002/2023/AKP.

REFERENCES

Kurt Akeley. 1993. Reality engine graphics. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. 109–116.

Tomas Akenine-Mo, Eric Haines, Naty Hoffman, et al. 2018. Real-time rendering. (2018).

AMD. 2022. FidelityFX Super Resolution 2.0. https://gpuopen.com/fidelityfx-superresolution-2/.

AMD. 2023. AMD FSR 3 Now Available. https://community.amd.com/t5/gaming/amd-fsr-3-now-available/ba-p/634265.

Dmitry Andreev. 2010. Real-time frame rate up-conversion for video games: or how to get from 30 to 60 fps for" free". In *ACM SIGGRAPH 2010 Talks*. 1–1.

Connelly Barnes, Eli Shechtman, Dan B Goldman, and Adam Finkelstein. 2010. The generalized patchmatch correspondence algorithm. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part III 11*. Springer, 29–43.

Huw Bowles, Kenny Mitchell, Robert W Sumner, Jeremy Moore, and Markus Gross. 2012. Iterative image warping. In *Computer graphics forum*, Vol. 31. Wiley Online Library, 237–246.

Karlis Martins Briedis, Abdelaziz Djelouah, Mark Meyer, Ian McGonigal, Markus Gross, and Christopher Schroers. 2021. Neural frame interpolation for rendered content. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–13.

Karlis Martins Briedis, Abdelaziz Djelouah, Raphaël Ortiz, Mark Meyer, Markus Gross, and Christopher Schroers. 2023. Kernel-Based Frame Interpolation for Spatio-Temporally Adaptive Rendering. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–11.

Robert L Cook. 1986. Stochastic sampling in computer graphics. *ACM Transactions on Graphics (TOG)* 5, 1 (1986), 51–72.

Andrew Edelsten, Paula Jukarainen, and Anjul Patney. 2019. Truly next-gen: Adding deep learning to games and graphics. In *Game Developers Conference*.

Epic. 2022. Temporal Super Resolution. https://docs.unrealengine.com/5.2/en-US/temporal-super-resolution-in-unreal-engine/.

Epic Games. 2023. *The most powerful real-time 3D creation tool - Unreal Engine*. https://www.unrealengine.com/en-US

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 Transactions on Graphics (TOG)* 40, 6 (2021), 1–16.

Johannes Hanika, Lorenzo Tessari, and Carsten Dachsbacher. 2021. Fast temporal reprojection without motion vectors. *Journal of Computer Graphics Techniques Vol* 10, 3 (2021).

Sam Holmes and Jonathan Wicks. 2020. QCOM Frame Extrapolation. https://registry.khronos.org/OpenGL/extensions/QCOM/QCOM_frame_extrapolation.txt

Yinlin Hu, Rui Song, and Yunsong Li. 2016. Efficient coarse-to-fine patchmatch for large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5704–5712.

Jorge Jimenez, Jose I Echevarria, Tiago Sousa, and Diego Gutierrez. 2012. SMAA: Enhanced subpixel morphological antialiasing. *Computer Graphics Forum* 31, 2pt1 (2012), 355–364.

Khronos. 2022. OpenGL 4 Reference Pages - imageAtomicMin. https://registry.khronos.org/OpenGL-Refpages/gl4/html/imageAtomicMin.xhtml.

Sungkil Lee, Younguk Kim, and Elmar Eisemann. 2018. Iterative depth warping. *ACM Transactions on Graphics (TOG)* 37, 5 (2018), 1–13.

Zhan Li, Carl S Marshall, Deepak S Vembar, and Feng Liu. 2022. Future Frame Synthesis for Fast Monte Carlo Rendering. In *Graphics Interface*.

Edward Liu. 2020. DLSS 2.0-Image Reconstruction for Real-time Rendering with Deep Learning. In *GPU Technology Conference (GTC)*.

William R Mark, Leonard McMillan, and Gary Bishop. 1997. Post-rendering 3D warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*. 7–16.

NVIDIA. 2023. DLSS 3: AI-Powered Neural Graphics Innovations. https://www.nvidia.com/en-sg/geforce/news/dlss3-ai-powered-neural-graphics-innovations/.

Qualcomm. 2023. Mobile Gaming. https://developer.qualcomm.com/solutions/mobile-gaming

Alexander Reshetov. 2009. Morphological antialiasing. In *Proceedings of the Conference on High Performance Graphics 2009*. 109–116.

Iain E Richardson. 2004. *H. 264 and MPEG-4 video compression: video coding for next-generation multimedia*. John Wiley & Sons.

Andre Schollmeyer, Simon Schneegans, Stephan Beck, Anthony Steed, and Bernd Froehlich. 2017. Efficient hybrid image warping for high frame-rate stereoscopic rendering. *IEEE Transactions on Visualization and Computer Graphics* 23, 4 (2017), 1332–1341.

Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. 1998. Layered depth images. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. 231–242.

Manu Mathew Thomas, Karthik Vaidyanathan, Gabor Liktor, and Angus G Forbes. 2020. A reduced-precision network for image reconstruction. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–12.

Unity. 2023. Unity Engine. https://unity.com/products/unity-engine

Songyin Wu, Sungye Kim, Zheng Zeng, Deepak Vembar, Sangeeta Jha, Anton Kaplanyan, and Ling-Qi Yan. 2023a. ExtraSS: A Framework for Joint Spatial Super Sampling and Frame Extrapolation. In *Proceedings of the 2023 SIGGRAPH Asia Conference*. 1–11.

Zhizhen Wu, Chenyu Zuo, Yuchi Huo, Yazhen Yuan, Yifan Peng, Guiyang Pu, Rui Wang, and Hujun Bao. 2023b. Adaptive recurrent frame prediction with learnable motion vectors. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11.

Lei Xiao, Salah Nouri, Matt Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. 2020. Neural supersampling for real-time rendering. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 142–1–12.

Xiangyu Xu, Li Siyao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. 2019. Quadratic video interpolation. *Advances in Neural Information Processing Systems* 32 (2019).

Lei Yang, Shiqiu Liu, and Marco Salvi. 2020. A survey of temporal antialiasing techniques. *Computer Graphics Forum* 39, 2 (2020), 607–621.

Lei Yang, Yu-Chiu Tse, Pedro V Sander, Jason Lawrence, Diego Nehab, Hugues Hoppe, and Clara L Wilkins. 2011. Image-based bidirectional scene reprojection. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. 1–10.

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.

Zheng Zeng, Shiqiu Liu, Jinglei Yang, Lu Wang, and Ling-Qi Yan. 2021. Temporally Reliable Motion Vectors for Real-time Ray Tracing. *Computer Graphics Forum* 40, 2 (2021), 79–90.

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*. 1–10.

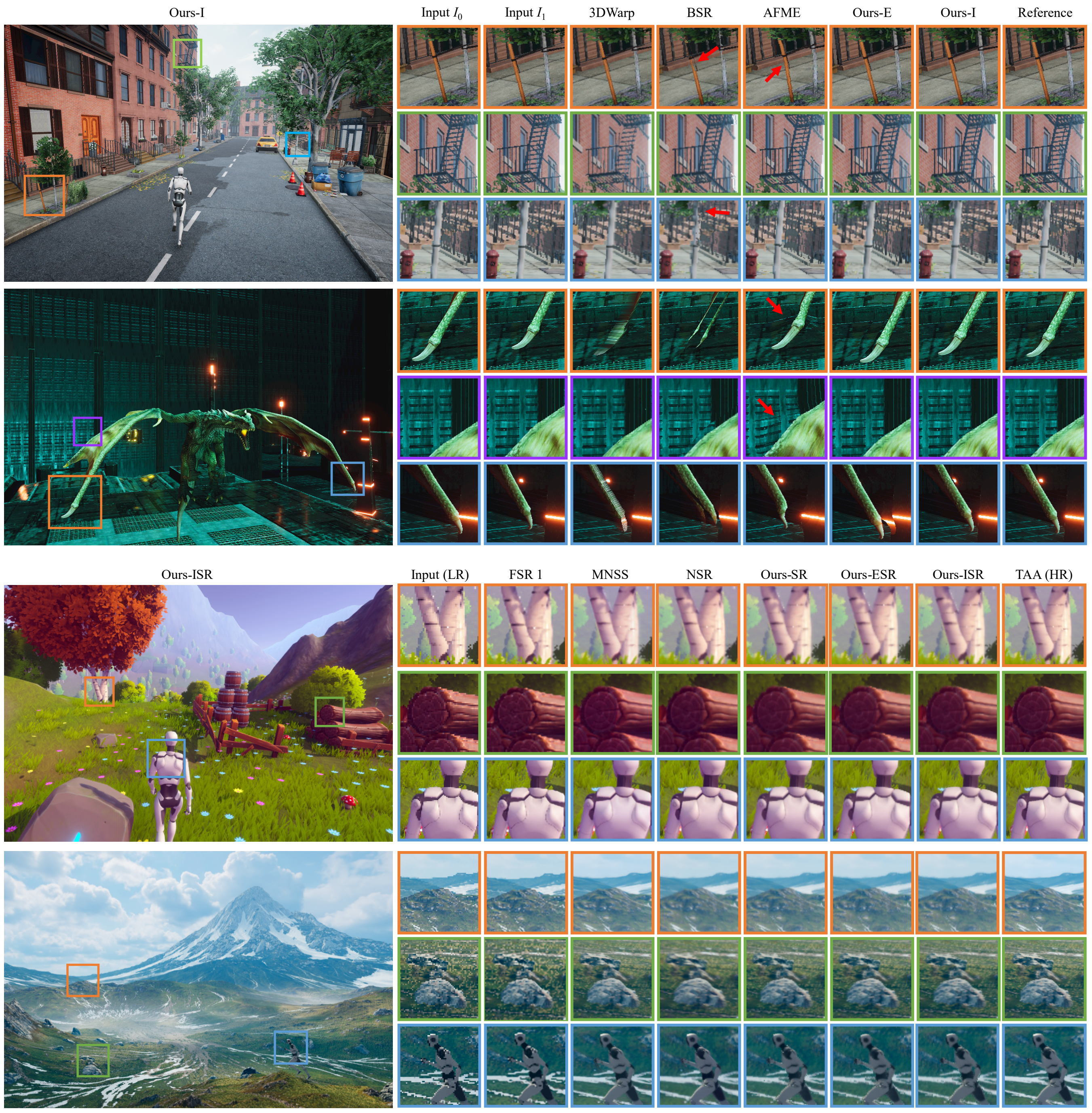

Figure 7: Comparison of our method to supersampling baselines in Unity scenes, including frame generation methods (3DWarp, BSR, AFME) and SR methods (FSR 1, MNSS, NSR). MSAA and TAA serve as frame generation and SR references, respectively. In frame generation (upper section), the Ours-I model provides accurate motion estimation while maintaining high image quality. The Ours-E model also produces comparable quality, but it occasionally deviates in motion estimation (see the dragon's wingtip in the orange box) and produces artifacts in disocclusions (see the right region of dragon's wingtip in the blue box). Baseline methods have several limitations, including 'rubber sheet' artifacts in 3DWarp, motion loss of thin objects in BSR, and frequent distortions in AFME. In SR (lower section), the Ours-SR, Ours-ISR, and Ours-ESR models consistently produce high-quality results, trailing deep learning-based methods MNSS and NSR but outperforming FSR 1.

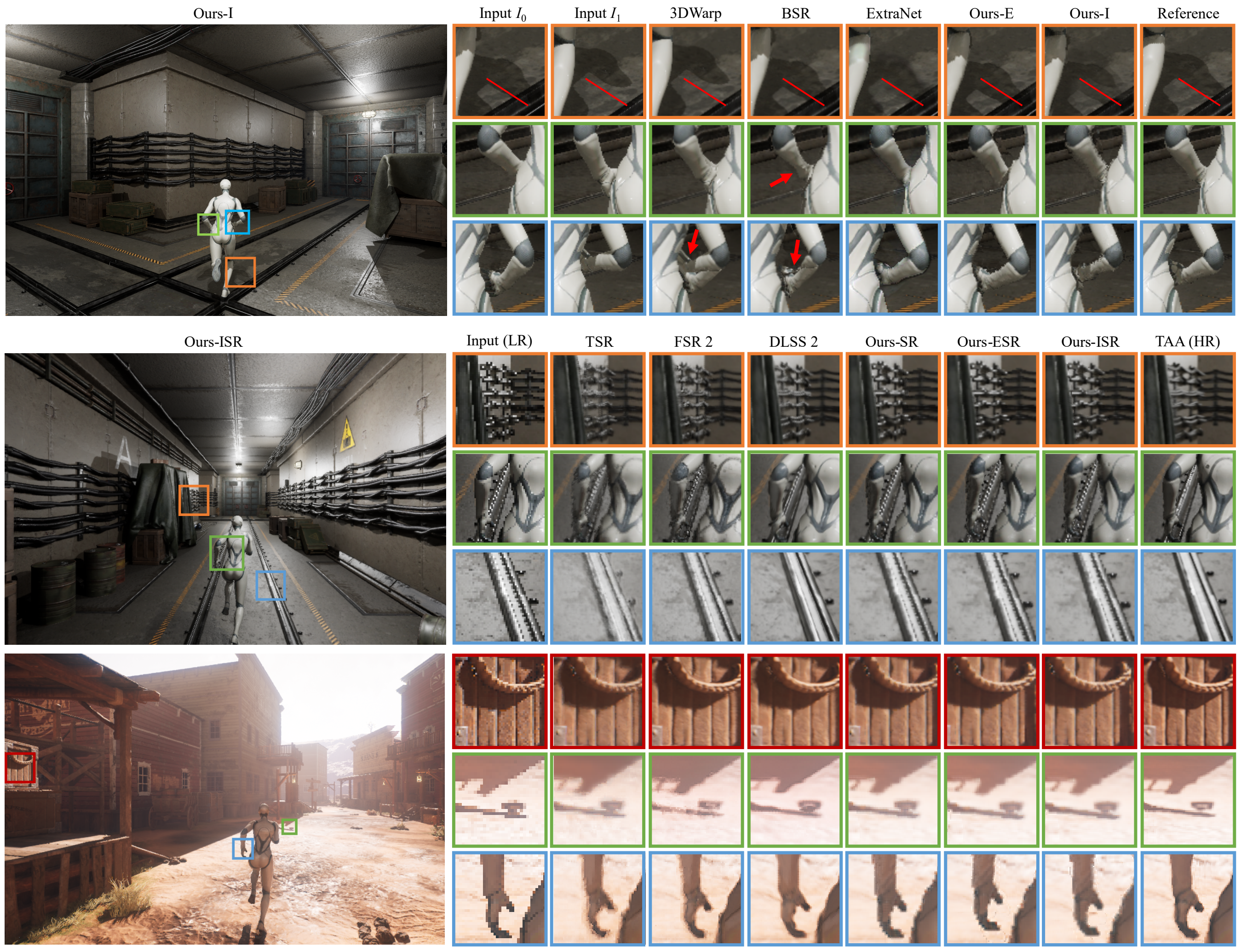

Figure 8: Comparison of our method to supersampling baselines in UE scenes, including frame generation methods (3DWarp, BSR, ExtraNet) and SR methods (TSR, FSR 2, DLSS 2). In frame generation (upper section), ExtraNet employs a deep learning model to predict shading changes, offering more accurate shadow predictions (see red line indicators). On the contrary, other lightweight methods do not effectively handle dynamic shadows. In the SR task (lower section), TSR, FSR2, and our method all produce comparable improvements in image resolution. However, they have difficulty with the edges of fast-moving objects. In contrast, the deep learning approach (DLSS 2) generates smoother edges, resulting in higher visual quality.

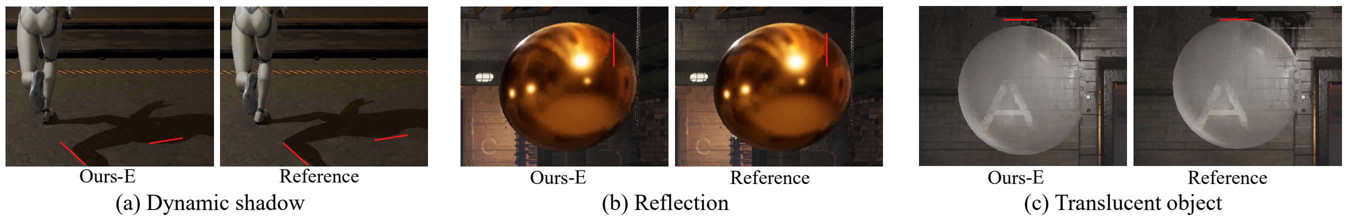

Figure 9: Example results of our extrapolation method on dynamic shadow (a), metal reflection (b), and a translucent object (c), with each compared to the reference image for clearer visibility. Our lightweight approach ignores costly shading calculations, which can cause low frame rate effects such as misaligned shadows, reflections, and translucent objects comparing with the reference (refer to red line indicators). Avoiding low frame rate inputs before frame generation can mitigate these issues. We recommend an input frame rate greater than 30 fps.