Motion Vector-Based Frame Generation for Real-Time Rendering

Inwoo Ha[1,2] Young Chun Ahn[1] Sung-eui Yoon[2]

[1]SAIT (Samsung Advanced Institute of Technology), South Korea
[2]KAIST (Korea Advanced Institute of Science and Technology), South Korea

Figure 1: *Our frame interpolation method features motion and image context encoder-decoder, outperforming state-of-the-art neural network techniques, including IFRNet [KJL*22] and Mob-FGSR [YZZ*24], in preserving fine geometric and texture details.*

Abstract

The demand for high frame rate rendering is rapidly increasing, especially in the graphics and gaming industries. Although recent learning-based frame interpolation methods have demonstrated promising results, they have not yet achieved the quality required for real-time gaming. High-quality frame interpolation is critical for rendering faster, dynamic motion during gameplay. In graphics, motion vectors are typically favored over optical flow due to their accuracy and efficiency in game engines. However, motion vectors alone are insufficient for frame interpolation, as they lack bilateral motions for the target frame to interpolate and struggle with capturing non-geometric movements. To address this, we propose a novel method that leverages fast, low-cost motion vectors as guiding flows, integrating them into a task-specific intermediate flow estimation process. Our approach employs a combined motion and image context encoder-decoder to produce more accurate intermediate bilateral flows. As a result, our method significantly improves interpolation quality and achieves state-of-the-art performance in rendered content.

CCS Concepts

• ***Computing methodologies*** → ***Image-based rendering;***

1. Introduction

Significant progress has been made in rendering high quality computer generated images, which are now widely used in movies, animations, AR/VR applications, and video games. However, the growing demand for photorealistic quality presents the challenge of achieving real-time performance with high frame rates. Frame interpolation methods, which involve quickly computing a subset of additional in-between frames from an existing set of frames, can greatly alleviate this computational burden. By reducing the num-

DOI: 10.2312/pg.20251297

ber of frames that need to be rendered, both rendering times and costs are significantly decreased.

Recently, deep learning-based methods [Liu20; XNC*20; BDM*21; BDO*23] have gained attention for achieving promising results in rendered content. These techniques benefit from auxiliary information readily available in the rendering pipeline. In particular, motion vectors, which provide precise values on relative movements of an object point compared to the previous frame, have been highly effective in addressing superresolution challenges, leading to the development of commercial solutions like DLSS [Liu20], AMD FSR, and Intel XeSS. This motion vector can be easily applied to frame generation like recent NVIDIA DLSS3 and AMD FSR3, which are prominent commercial applications. However, motion vectors lack non-geometric information, such as shadows, lighting changes, particles, transparency and texture animations, all of which are crucial for rendering frames. Additionally, due to the nonlinear nature of motions and the numerous errors caused by occlusions and disocclusions, directly employing motion vectors for frame interpolation results in significant misalignments and errors, making their use for frame interpolation ambiguous.

Recent studies have investigated optical flow estimation for frame generation alongside the use of rendered motion vectors. Interpolation methods, such as learnable motion vectors [WZH*23], NFI [BDM*21], and KBI [BDO*23], rely on estimated optical flow or motion vectors to generate intermediate frames, assuming that the motion vector for the target frame has already been rendered. This approach enables the derivation of precise bilateral motion vectors. However, this approach complicates the integration with existing interfaces in current game engines and incurs significant overhead, as it requires rendering an additional G-buffer for the target frame to interpolate.

Meanwhile, Mob-FGSR [YZZ*24] streamlines the process by utilizing only motion vectors from key frames as input, similar to commercial solutions like DLSS3 [Liu20] and AMD FSR3, enabling seamless integration with existing interfaces in current game engines. The method refines these input motion vectors into approximate bilateral flows, which are then enhanced using heuristic inpainting and image synthesis methods. However, this approach struggles with large motions and disocclusions, leading to increased errors in the estimated motion vectors and requiring substantial inpainting. Furthermore, it fails to account for non-geometric effects, instead blending them directly from key frames—often resulting in visually blurry outputs.

Video Frame Interpolation (VFI) focuses on generating intermediate frames from a pair of input frames. Among various approaches, flow-based methods have gained considerable traction and shown impressive performance in recent works [HZH*22; LWL*22; JSJ*18; KJL*22; LZH*23; RKT*22]. These methods typically estimate optical flows between the input frames, warp the frames and their contextual features accordingly, and then synthesize the intermediate frame from the warped representations. Consequently, the quality of the output frame is highly dependent on the accuracy of the flow estimation. Most existing flow-based VFI methods compute bilateral flows using pyramid-based architectures. While effective for moderate motion, the limited depth of these pyramids constrains their receptive field, making it challenging to accurately capture large or complex motions. Furthermore, optical flow estimation often becomes unreliable when dealing with rapid, large-scale movements or small, fast-moving objects—situations that are especially common in gaming environments. These limitations lead to instability and decreased reliability of optical flow-based methods in real gaming environments, especially under dynamic camera movements and complex object interactions.

To overcome these issues, we propose a novel approach that refines approximate bilateral motion vectors by incorporating image context, estimates intermediate residual flows, and synthesizes frames by merging these flows. Our approach is not trivial. By leveraging both motion vectors and image context, the network intelligently interprets the input approximate motion vectors, effectively handling large motions and disocclusions while preserving fine details from the initial coarse stage. Our ablation study further demonstrates that naive combinations of motion vectors and residual flows are insufficient, emphasizing the value and effectiveness of our proposed method.

Our design is based on the observation that the commonly used UNet-like pyramid architecture [KJL*22] tends to accumulate errors in its early coarse stages, especially when dealing with large and complex motions. Intermediate flows generated solely by image feature decoders are generally effective at handling small motions but often struggle with large motions, especially during the early stages. On the other hand, precomputed approximated flows, though rough, imprecise, and not explicitly task-oriented, can assist in reasoning during the initial coarse stages and enhance the overall flow quality. To address these challenges, we propose a strategy that incorporates motion priors, such as approximated motion vectors for the target frame, as input and utilizes them from the early stages of processing. Our approach has demonstrated high effectiveness in managing large motions and fast-moving small, thin objects.

By analyzing and integrating these guiding flows with the pyramid encoder, the network can better maintain the fidelity of motion flows across different scales, thereby establishing a robust foundation for subsequent refinement processes. This integration not only improves the initial motion estimation but also ensures that the finer details are preserved throughout the processing stages, leading to more accurate and reliable outcomes. So the network can focus on regions that are more efficient and accurate, enabling motion vector and intermediate residual flows to work together synergistically.

With the proposed approach, our experiments demonstrate the robust performance of our method on the public gaming scenes, notably outperforming SOTA techniques in scenarios involving large motions and dynamic small objects. We also show the versatility of our approach by extending it to handle general flows alongside motion vectors.

2. Related Works

2.1. Motion Vector-Based Approaches

Recently, demands for high quality games necessitates high-frame-rate rendering, where motion vectors play a crucial role across most

graphics applications. Approaches like TAAU [YLS20] and AMD FSR employ heuristic techniques, utilizing handcrafted weights to seamlessly integrate low-resolution jittered images with warped high-resolution images from previous outputs, yielding high resolution results. Supersampling like DLSS 2.0 [Liu20] and Intel XeSS, incorporating reprojection with motion vectors, have been introduced and embraced within the industry. This approach involves leveraging rendering outputs from the previous frame for rendering the current frame, consequently significantly decreasing the average rendering workload. Moreover, in [GFL*21], temporal oversampling is attained by utilizing the G-buffer as input and conducting frame extrapolation. [ZLY*21] present a technique to estimate motion vectors capable of tracking shadows and glossy reflections in real-time. Learnable motion vectors [WZH*23], ExtraSS [WKZ*23], NFI [BDM*21], and KBI [BDO*23] leverage motion vectors to produce intermediate frames, assuming that the motion vector for the target frame has already been rendered. However, this method requires significant adaptation of the renderer interfaces and add significant overhead of additional G-buffer rendering for the target frame. Recently, Mob-FGSR [YZZ*24] proposes lightweight frame generation framework suitable for mobile real-time rendering without the need of the motion vector for the target frame. It share motion vector refinement based on splat with ours, but they are not deep learning based approach and rely entirely on motion vector and heuristic image synthesis method.

The rendering approaches mentioned above employ motion vectors to warp previous or reconstructed frames onto the current frame, with these vectors precisely defining geometric movements. While utilizing estimated motion vectors seems to be the most straightforward method for frame interpolation, the absence of the precise bilateral motion vectors for frame interpolation is problematic.

2.2. Video Frame Interpolation

Video Frame Interpolation represents an active area of research that has greatly profited from advancements in deep learning, showing notable interpolation results in recent studies. Due to the reliability of optical flow, flow-based techniques have become prominent in VFI [NL18; XSS*19; JSJ*18; PKLK20; NL20; KJL*22; XCW*19; RKT*22] and rendering pipelines [BDM*21; BDO*23; Liu20]. Previous methods using optical flow can be divided into two categories. The first category involves a pretrained flow model combined with an independent synthetic network [NL18; XSS*19; JSJ*18; PKLK20; NL20]. In this approach, the pretrained flow model estimates flows and synthesis images independently, demonstrating promising results. However, the two-step pipeline can indeed overlook the disparity between true optical flow and task-specific objectives [XCW*19; LYT*17], potentially resulting in suboptimal performance for a given task. Our approach, while also utilizing independently estimated flows, addresses this issue by employing them only as a form of guidance rather than as strict constraints. Notably, our model does not necessitate high-quality optical flows; even fast and inexpensive flows, such as motion vectors approximated to the target frame, are sufficient. Despite this, our approach achieves significantly better quality in frame interpolation, demonstrating the efficacy of using approximate flows for guidance .

Another category involves a jointly trained estimation module to derive flow estimation [LZH*23; KJL*22; XCW*19; RKT*22]. In this approach, bilateral flows and the interpolated intermediate features are updated jointly, eliminating the need for an independent synthetic network. These methods demonstrate promising interpolation quality by directly predicting task-oriented flows from the coarsest level. However, they struggle with modeling large motions and small objects due to their limited receptive field. Our approach addresses these challenges by effectively handling large motions through the integration of independently estimated flows into task-oriented intermediate flow estimation.

2.3. Optical Flow Estimation

FlowNet [FDI*15] demonstrates that convolutional neural networks can predict per-pixel optical flow fields through artificially generated datasets for optical flow estimation, employing the encoder-decoder U-shaped network architecture. SPyNet [RB17] introduces the spatial pyramid approach, while PWC-Net [SYLK18] incorporates feature warping and computes cost volume. Fast-FlowNet [KSY21] integrates pyramid features and backward warping, while LiteFlowNet [HTL18] applies iterative refinement using coarse-to-fine pyramids. Unlike conventional pyramid-based approaches, RAFT [TD20] constructs 4D correlation volumes for flow computation. The AMT [LZH*23] enhances the all-pairs correlation within RAFT [TD20], effectively capturing dense correspondence between frames.

3. Method

Given two consecutive input frames I_0 and I_1 rendered at adjacent time instances, the objective of our frame interpolation is to predict their intermediate frame I_t, where $0 < t < 1$, by leveraging information from both frames I_0 and I_1 and their guiding flows $M_{t\to 0}$ and $M_{t\to 1}$, such as motion vectors generated by the rendering pipeline.

As shown in Fig. 2, our design is composed of two main components: one for motion context pyramid, and the other for image context pyramid. Within the image context pyramid, intermediate features and flows undergo progressive refinement from the input images. Concurrently, in the motion context pyramid, motion priors extracted from motion vectors are refined into multi-scale task-oriented flows and flow features. These refined flows and features are then injected into the image context pyramid decoders starting from its early coarse level. This integration improves the generation of accurately aligned task-oriented intermediate flows across various scales. This approach not only enhances the accuracy of early motion estimation but also maintains finer details across all levels of decoders, leading to more reliable outcomes. Consequently, we can infer information from high level motion such as camera movement to local motions such as small object movement.

Specifically, the motion context module warps both input images to the frame I_t using their refined motion vectors. A motion feature encoder is then used to extract multi-scale guiding flow features at each level of the pyramid from the t-aligned input images and their

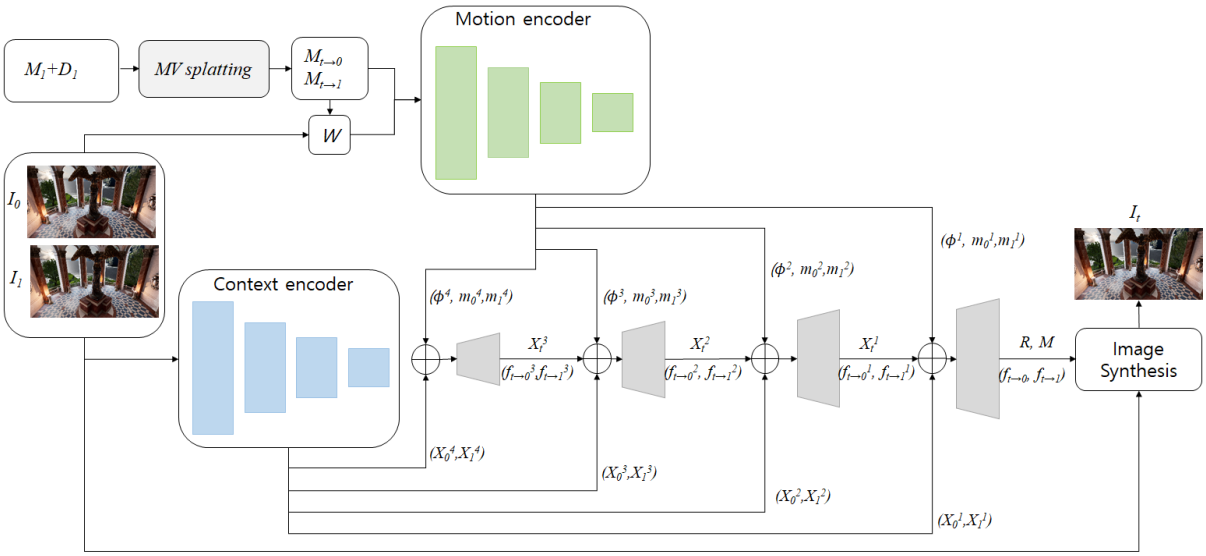


Figure 2: *Architecture overview of the proposed method. The overall architecture consists of a preprocessing step and two distinct feature pyramids. Initially, during preprocessing, the input motion vector M_1 is splatted onto its predicted pixel positions for the target frame I_t. Then, using the splatted motion vectors, the input images are warped to align with the target frame I_t. Subsequently, the warped input frames and refined motion vectors are fed into the motion encoding pyramid, which produces refined guiding flows and multi-level flow features. Meanwhile, the input frames are processed by the image context encoder, yielding multi-level image features. Afterwards, the generated flow and image features are input into the combined decoder, which simultaneously updates both intermediate features and flows by merging the refined guiding flows with the intermediate flows. Finally, we synthesize the intermediate frame using the final estimated flow fields, occlusion masks, and residuals.*

corresponding refined motion vectors. And the image context pyramid uses a feature encoder to extract multi-scale image features for each input frame. Once these hierarchical motion and image context features are obtained, we progressively refine the intermediate flows and features using multi-level decoders. During this refinement, the extracted multi-scale image context features are warped to align with the target frame I_t, guided by the corresponding flow fields that integrate refined guiding flow and intermediate flows. Finally, a frame synthesis module is employed to generate the refined intermediate frame.

3.1. Motion Context Encoder

We use the motion vector and depth image provided by the rendering engine as the input. While other applications, such as super sampling [Liu20; XNC*20; MES*23], leverage the motion vector for precise re-projection due to its calculation from exact geometries for the target frame, in frame interpolation, we can only obtain the motion vector M_1 and depth D_1 from the rendered reference frame I_1 instead of the target intermediate frame I_t.

When the motion vectors $M_{t\to0}$ and $M_{t\to1}$ for the target frame I_t are not provided through deferred rendering, our method begins by estimating the motion vectors based on the available keyframe data. Following the approach of [YZZ*24], we employ a fast splat-based motion estimation technique that leverages depth and motion vectors from rendered keyframes, both of which are included in the G-buffer provided by game engines in real-time, to produce bilateral motion vectors for the intermediate frames. However, we assume linear motions rather than quadratic motions, and disocclusions are filled using motion vectors from the key frames, leveraging the capabilities of neural networks.

With the motion vector M_1 in image space as input, we compute the positions P_t for the desired interpolation time t. The pixel motion in the generated frames is then determined by calculating the bilateral motion vectors $M_{t\to0}$ and $M_{t\to1}$. The construction of motion vectors proceeds by embedding the calculated vectors into their corresponding pixels. In particular, $M_{t\to1}$ is assigned to the position P_t through splatting [SGHS98]. This splatting process is applied to each pixel in keyframe I_1, producing the projected motion vectors $M_{t\to0}$ and $M_{t\to1}$. Assuming locally linear motion, we estimate the motion of each pixel in the adjacent frames using the formula $P_t = P_1 - (1-t)\cdot M_1$, where t represents the time of frame generation, and P_t denote the predicted pixel positions for the interpolated frame.

To resolve conflicts during the splatting process, where multiple vectors may be projected onto the same pixel, we use motion vector splatting based on depth comparison with the input buffer D_1. This approach ensures that, in cases of conflict, the motion vectors with the smallest depth are preserved. For disocclusion filling, we efficiently approximate the gaps in the motion vectors by directly using the vectors from M_1.

While the splatting process provides more accurate motion vectors, they are still inadequate for precise frame interpolation. Spe-

cially, mismatches between surface pixels occur for several reasons, including incorrect assumptions about locally linear motions, negating the vector direction for $f_{t\to1}$ without considering occlusions, actual motions not being captured by the motion vectors, and the heuristic approach used for disocclusion filling. These mismatches result in inaccurate warping and unreliable updates of intermediate flows and features, and this problem becomes particularly significant with large motions. Consequently, instead of using the spatted motion vectors directly, we first refine the bilateral motion vectors with the help of input image pairs.

To refine the input guiding flows, we opt a coarse-to-fine manner, which process is necessary to generate a faithful intermediate features. As a preprocessing step, each input image, I_0 and I_1, is warped to the t frame using its guiding flow, $M_{t\to0}$ and $M_{t\to1}$. This alignment enables the input images to match the guiding flow defined in the t frame. As a result, we anticipate that the encoder will extract contextual information, such as confidence values related to motion context from the motion vector, based on the context of the aligned images I_0 and I_1. This process is repeated for the other input frame, and the resulting images are concatenated with the first frame image and guiding flows. Then they are fed into a pyramid encoder, which generates multi-level refined guiding flow features and guiding flows $\{\phi_t^k, m_0^k, m_1^k | k \in \{1,2,3,4\}\}$, as shown in Fig 2.

The pyramid encoder consists of four convolutional blocks, and the encoded guiding flow feature ϕ_t^k is defined as:

$$\phi_t^k = E_m(W(I_0, M_{t\to0}), W(I_1, M_{t\to1}), M_{t\to0}, M_{t\to1}) \quad (1)$$

where W denotes the backward warping operator, and E_m represents the encoder function applied to the input images and their corresponding guiding flows, for each pyramid level $k \in 1,2,3,4$. Each encoder block consists of two 3×3 convolutional layers with strides of 2 and 1, respectively. The number of feature channels at pyramid levels 1 through 4 is set to 32, 48, 72, and 96, respectively. This hierarchical encoding yields multi-scale guiding flow features, allowing the network to better preserve motion fidelity across scales. As a result, the encoded features provide a robust foundation for motion refinement, enhancing both initial flow estimation and the preservation of fine details throughout the pipeline, leading to more accurate and reliable interpolation results.

3.2. Frame Interpolation

To obtain a coarse-to-fine contextual representation from each input image, we utilize the image pyramid encoder E_I, which extracts a pyramid of features. This encoder processes the two input images I_0, I_1 to produce the pyramid of image features $\{X_0^k, X_1^k | k \in \{1,2,3,4\}\}$. At each pyramid level, the encoder performs two convolutions with down-sampling. Our network extracts four levels of pyramid features to facilitate further progressive warping.

After extracting hierarchical features of both motions and images, we progressively refine the intermediate flows and intermediate features using 4 level pyramid decoders to generate the intermediate features X_t^k and the intermediate flows $f_{t\to0}^k$, $f_{t\to1}^k$. Concretely, the features ϕ_t^k and flows m_0^k, m_1^k extracted from guiding flow encoder are injected into a pyramid decoder, which generates multi-level refined bilateral guiding flows $f_{t\to0}^k, f_{t\to1}^k$ and intermediate features X_t^k. Then, features among decoders can be computed by

$$[f_{t\to0}^3, f_{t\to1}^3, X_t^3] = D_f^4([X_0^4, X_1^4, \phi^4, m_0^4, m_1^4, T]), \quad (2)$$

$$[f_{t\to0}^{k-1}, f_{t\to1}^{k-1}, X_t^{k-1}] = D_f^k([X_0^k, X_1^k, X_t^k, \phi^k, m_0^k, m_1^k]), \quad (3)$$

$$[f_{t\to0}, f_{t\to1}, R, M] = D_f^1([X_0^1, X_1^1, X_t^1, \phi^1, m_0^1, m_1^1]) \quad (4)$$

Here, D_f^k $(k = 1,2,3,4)$ denotes the pyramid decoders, each consisting of three stages: flow merging, warping, and a decoding network. First, the generated intermediate flows $f_{t\to0}^k$ and $f_{t\to1}^k$ are combined with refined guiding flows from the guiding flow encoder to produce merged bilateral flows $\bar{f}_{t\to0}^k$ and $\bar{f}_{t\to1}^k$, using $\bar{f}^k = m^k + f^k$. Next, the decoder performs backward warping on multi-level image context features X_0^k and X_1^k using the merged flow fields $\bar{f}_{t\to0}^k$ and $\bar{f}_{t\to1}^k$. The warped image context features $\bar{X}_0^k$, $\bar{X}_1^k$, intermediate features X_t^k, motion context features ϕ^k, and flows m_0^k, m_1^k are then passed to the decoding network. As the decoding network, we adopt the IFRBlock decoder from IFRNet [KJL*22], where each layer consists of six convolutional layers and one deconvolution layer, with strides of 1 and 1/2, respectively. A single-channel conditional input T, filled with the target time value t, is introduced to enable arbitrary time interpolation. Although T is implicitly encoded through motion vector splatting and relative warping, we explicitly incorporate it into the first decoder to handle residual motions not accounted by motion vectors. This explicit temporal conditioning helps guide the synthesis of the target frame more accurately.

In flow-based VFI methods [JSJ*18], the common formulation for interpolating the final intermediate frame is:

$$I_t = M \cdot W(I_0, f_{t\to0}) + (1-M) \cdot W(I_1, f_{t\to1}) + R, \quad (5)$$

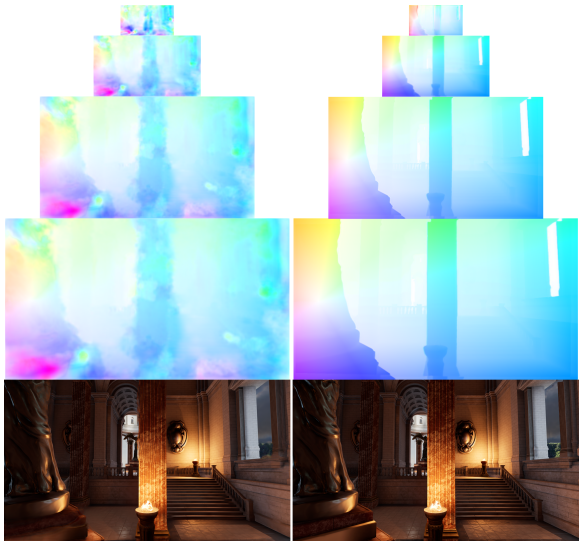
where W is the warping operator and $\cdot$ is the element-wise multiplication operator. M is a one channel merge mask exported by a sigmoid layer whose elements range from 0 to 1, and R is a three-channel image residual that can compensate for details. $f_{t\to0}$ and $f_{t\to1}$ are final predicted values of the bilateral flows. Finally, we can synthesize the desired frame I_t.

3.3. Loss Functions

Our method utilizes two types of losses for supervising image reconstruction between the generated frame It and the ground truth frame I_t^{gt}. We use the Charbonnier loss [CBAB94], a smooth and more robust variant of the L2 loss, defined as $\mathcal{L}_{char}(x) = (x^2+\epsilon^2)^\alpha$, and the census loss $\mathcal{L}_{cen}$ [MHR18], which computes a soft Hamming distance between census-transformed patches of I_t and I_t^{gt}. The census loss enhances robustness to illumination changes and outliers by relying on structural similarity and relative intensity patterns. The combined loss is formulated as:

$$\mathcal{L} = \lambda_{char}\mathcal{L}_{char} + \lambda_{cen}\mathcal{L}_{cen} \quad (6)$$

where λ_{char} and λ_{cen} are the corresponding loss weights. In our implementation, both are set to 1.



IFRNet Ours

Figure 3: *We display flow pyramids and generated images for IFRNet[KJL*22] and our model, demonstrating that our model effectively captures fine flow details starting from the coarsest level. This capability allows our model to leverage cues from the coarse level to guide the identification of fine details.*

4. Experiments

To demonstrate the effectiveness and efficiency of our method in rendered frame prediction, we compare both its quality and inference speed against state-of-the-art frame interpolation approaches. Additionally, extensive ablation studies are conducted to evaluate the impact of each proposed component. All experiments follow the common protocol of setting $t = 0.5$, corresponding to the single-channel conditional input $T = 0.5$, with motion vectors preprocessed via splatting at the mid-frame position.

4.1. Training Details

We train our method on our created training set for 300 epochs using the AdamW optimizer [LH17] across four NVIDIA A6000 GPUs. The total batch size is 64, and the learning rate follows a cosine attenuation schedule, starting at 5×10^{-4} and gradually decreasing to 1×10^{-5}. We apply data augmentation, which includes random flipping, rotating, and cropping patches of size 224×224. Both parameters λ_{char} and λ_{cen} are set to 1.

4.2. Dataset

To evaluate the effectiveness of our algorithm in handling large and dynamic motions under realistic gaming conditions, we constructed a large-scale dataset using seven publicly available scenes from the Unreal Engine (UE) Marketplace. These scenes simulate diverse and complex gaming environments, featuring a wide range of shading effects such as glossy reflections, soft shadows, multiple light sources, dynamic camera movements, and intricate occlusions.

Sequential frames were rendered using Unreal Engine 4 at 1080p resolution and saved to disk, each comprising a rasterized image, motion vector, and depth buffer. To ensure high-quality, alias-free ground truth images, we employed 9× MSAA during data generation. Motion vectors and depth information were obtained via custom compute shaders integrated into the UE4 render pipeline. Unlike color images used as ground truth for the predicted frame, the motion vector and depth images were generated by skipping one frame at a time. This is due to the limitation that intermediate frame G-buffers are not available in standard commercial rendering workflows. For each scene, we generated multiple sequences with varying camera viewpoints. Each training and testing sequence contains 200 image triplets, where each triplet includes three consecutive frames and a G-buffer associated with the third frame. The dataset splits and additional statistics are provided in Table 1.

Table 1: *Statistics of the training and testing dataset used in our experiments.*

Scenes	Training Sequences	Testing Sequences	Training Frames	Testing Frames
Abandoned Apartment	0	2	0	400
Factory	8	4	1600	800
Infiltrator	12	7	2400	1400
Subway Sequencer	0	4	0	800
Subway Train	4	2	800	400
Sun Temple	4	3	800	600
Zen Garden	5	0	1000	0

Table 2: *Quantitative comparison with state-of-the-art methods on the dataset we created. The best result is marked in bold, and the second best is underlined.*

Method	Use DL	Model (PSNR/SSIM)	Params (M)	Time (ms)
IFRNet-S [KJL*22]	Y	33.54/0.959	2.8	18
IFRNet-B [KJL*22]	Y	33.85/0.960	5.0	22
IFRNet-L [KJL*22]	Y	34.37/0.962	19.7	49
FILM [RKT*22]	Y	34.05/0.961	-	-
AMT-S [LZH*23]	Y	33.70/0.958	3.0	38
AMT-L [LZH*23]	Y	34.08/0.960	12.9	75
AMT-G [LZH*23]	Y	34.43/0.963	30.6	164
MobFGSR[YZZ*24]	N	<u>35.17/0.979</u>	-	1
Ours	Y	**37.56/0.985**	5.4	24

4.3. Quantitative Comparisons

We compare our method with several state-of-the-art frame interpolation models, including AMT [LZH*23], IFRNet [KJL*22], FILM [RKT*22], and Mob-FGSR [YZZ*24], all retrained on our dataset. As shown in Table 2, our approach consistently outperforms both learning-based and non-learning-based methods in terms of quality and efficiency. Mob-FGSR, a non-learning-based method, achieves fast inference and reasonable quality (35.17 PSNR, 0.979 SSIM) by leveraging motion vectors. However, it does not compute optical flow and thus cannot handle disocclusion, large motion, or regions affected by shadows and reflections—areas where motion vectors

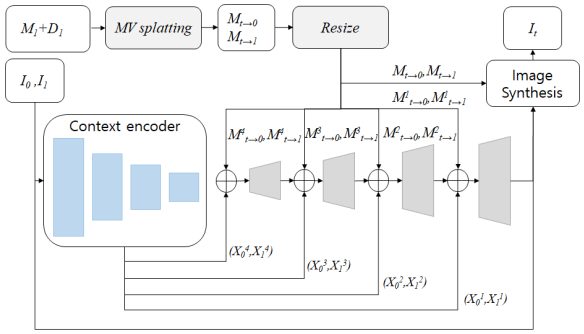


Figure 4: *An architecture variant without the motion encoder.*

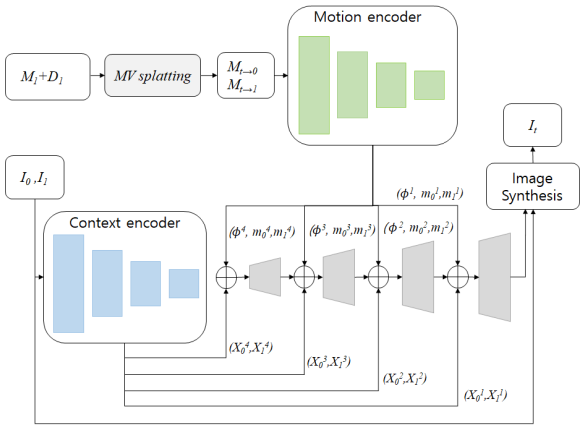


Figure 5: *An architecture variant that does not use aligned image input for the motion encoder.*

are incomplete or unavailable. This limitation leads to a noticeable quality gap compared to our method, which explicitly uses motion vectors within a deep learning framework and achieves superior performance (37.56 PSNR, 0.985 SSIM). Among learning-based models, our approach also outperforms large models such as AMT-G and IFRNet-L, exceeding them by more than 3.13 dB in PSNR and 0.022 in SSIM. Despite its smaller model size and lower computational cost, our method delivers higher visual fidelity. It processes 1080p frames in 24 ms using only 1.8 GB of GPU memory and achieves 11 ms per frame at 720p on an NVIDIA A6000 GPU. With continued advancements in hardware, we anticipate that further model optimization, lightweight design, and integration with supersampling will enable our method to run even faster, making it practical for real-time deployment in gaming environments.

4.4. Qualitative Comparisons

Figures 9 and 10 present qualitative comparisons of our frame interpolation results against state-of-the-art methods. Prior approaches often struggle to maintain the sharpness of moving object boundaries, particularly in the presence of large or complex motions. In contrast, our method more faithfully reconstructs motion boundaries and generates realistic textures with fewer artifacts, owing to its task-aware flow estimation design.

Table 3: *Ablation study on different architecture variants. The best result is marked in bold, and the second best is underlined.*

Architecture	PSNR/SSIM
Baseline (MV splat)	34.54/0.976
w/ External Flows	35.16/0.976
w/o Motion Encoder	35.42/0.974
w/o Aligned Input Images	<u>35.82/0.976</u>
Full Model	**37.56/0.985**

This performance gain stems from two key factors. First, by explicitly utilizing motion vectors, our method not only captures precise geometric motion but also effectively expands the receptive field during flow estimation. This enhances robustness to large displacements and allows the model to capture fine flow details even at coarse pyramid levels under large motion scenarios, as shown in Figure 3. Second, our method incorporates task-oriented optical flow, enabling it to handle appearance changes that are not represented by motion vectors—such as shadows, reflections, and lighting variations. As demonstrated in Figure 6, this contributes to more faithful reconstruction of such complex visual effects, resulting in improved overall image quality. Nevertheless, limitations remain, as illustrated in Figure 8. When faced with extreme scale variation or highly complex disocclusion—such as thin objects undergoing large displacement—the model may fail to inpaint missing regions accurately. These cases highlight the broader challenge of representing fine structures under extreme motion, where the neural capacity of the model becomes a limiting factor.

4.5. Ablation Study

To evaluate the effectiveness of the proposed methods, we conduct ablation studies focused on network architecture, as shown in Table 3.

Baseline As a baseline, we directly warp the input images I_0 and I_1 using the splatted input motion vectors $M_{t\to0}$ and $M_{t\to1}$, as outlined in Section 3.1. Disocclusions are filled using motion vectors from the key frames. The warped images are then blended with equal weights without any complicated image synthesis. It is important to note that the baseline has a PSNR that is 0.69 dB higher than IFRNet-B and 0.17 dB higher than IFRNet-L. This baseline highlights the effectiveness of the splatted input motion vectors.

External Optical Flows We conducted experiments using the off-the-shelf flow estimator LiteFlowNet [HTL18]. In these experiments, the input motion vector for the full model was replaced with the estimated flows. The model using external optical flow achieved a PSNR of 35.16 and an SSIM of 0.976. While these results exceeded those of state-of-the-art VFI methods, they were still lower than the performance obtained with approximated motion vector input. This implies our model to leverage cues from the coarse level to guide the identification of fine details with input external flows.

Without Motion Encoder To validate the effectiveness of our motion encoder, we eliminate the motion context encoder as shown in Fig. 4. Instead, the splatted motion vector for the target frame is supplied to each pyramid layer of the decoders, resized to cor-

Figure 6: *Handling non-motion vector effects. Changes in shadows and reflections caused by moving objects do not carry explicit motion vectors. Methods that rely solely on motion vectors tend to blend these regions across key frames without proper alignment, leading to blurry or inaccurate reconstructions. Our approach effectively captures and aligns such non-motion-vector effects—like changes in shadows and reflections—through a combined motion and context-aware strategy, enabling accurate synthesis of the target frame. Although IFRNet estimates optical flows, it still struggles to accurately reproduce these effects with ambiguous boundaries and gradual color transitions.*

respond with the resolution of each layer. Then the generated intermediate flows $f^k_{t\to 0}, f^k_{t\to 1}$ are combined with the motion vector $M^k_{t\to 0}$ and $M^k_{t\to 1}$ to produce the merged bilateral flows $\bar{f}^k_{t\to 0}$ and $\bar{f}^k_{t\to 1}$ using the formula $\bar{f}^k = M^k + f^k$. It is noteworthy that this architecture achieves a PSNR that is 0.88 dB higher than the baseline, yet 2.14 dB lower than the full model. This indicates that our guiding flow refinement process effectively enhances the input guiding flows. These results emphasize the effectiveness of the guiding flow refinement module in our architecture. The qualitative comparison for this ablation study is presented in Fig. 7.

Without Aligned Input Images in Motion Encoder To assess the effectiveness of aligned image input, we remove the aligned images from the motion context encoder, as shown in Fig. 5. In this setup, only splatted motion vectors are provided to the motion encoder without aligned images. It is important to note that this architecture achieves a PSNR that is 1.28 dB higher than the baseline, but 1.74 dB lower than the full model. This indicates that aligned image input plays a vital role, especially in large-motion scenarios. While unaligned input images may perform adequately under moderate motion, they fail under extreme conditions. A shallow pyramid network does not provide a sufficiently large receptive field to handle such large displacements effectively.

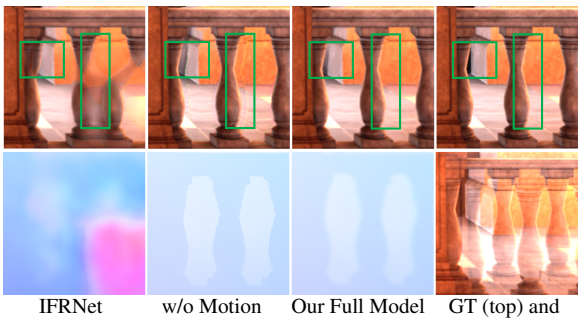


Figure 7: *Ablation study for motion encoder. IFRNet[KJL*22], our architecture variant without the motion encoder, and our complete model demonstrate the impact of the motion encoder, revealing more accurate flows (top) and improved image quality (bottom).*

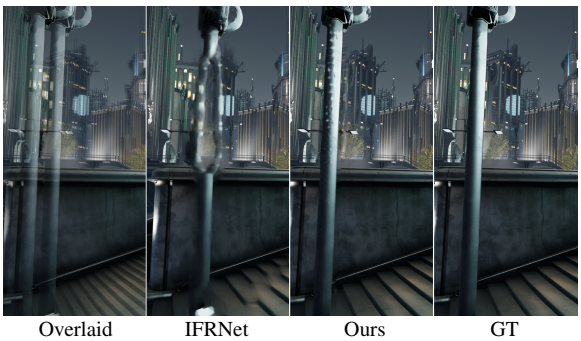


Figure 8: *Failure cases. While our method outperforms IFRNet [KJL*22], it still exhibits artifacts in challenging regions, such as around the pole with thin structures and large displacements. These issues arise from insufficient neural capacity to handle large scale variations and complex disocclusion scenarios. This highlights a fundamental challenge in representing fine-grained structures under highly dynamic motion.*

5. Conclusion

In this study, we proposed a novel method that leverages a motion context encoder to address the challenges of large motion in game frame interpolation. Our approach improves the quality of interpolation from the early stages of flow and feature estimation, achieving state-of-the-art performance on a large-motion benchmark. Nonetheless, limitations remain. Accurately interpolating fine details under large motion remains a challenging task, and our method is not immune to such difficulties. A promising direction for future work is analyzing the impact of different types of input guiding flows. While we assume access to motion vectors in game scenarios, it would be interesting to investigate how using block-matching-based flows—as might be available in more general settings—affects reconstruction quality. Furthermore, optimizing our model for deployment across various hardware platforms could enhance its practical value and broaden its applicability.

References

[BDM*21] BRIEDIS, KARLIS MARTINS, DJELOUAH, ABDELAZIZ, MEYER, MARK, et al. "Neural frame interpolation for rendered content". *ACM Transactions on Graphics (TOG)* 40.6 (2021), 1–13 2, 3.

[BDO*23] BRIEDIS, KARLIS MARTINS, DJELOUAH, ABDELAZIZ, ORTIZ, RAPHAËL, et al. "Kernel-Based Frame Interpolation for Spatio-Temporally Adaptive Rendering". *ACM SIGGRAPH 2023 Conference Proceedings*. 2023, 1–11 2, 3.

[CBAB94] CHARBONNIER, PIERRE, BLANC-FERAUD, LAURE, AUBERT, GILLES, and BARLAUD, MICHEL. "Two deterministic half-quadratic regularization algorithms for computed imaging". *Proceedings of 1st international conference on image processing*. Vol. 2. IEEE. 1994, 168–172 5.

[FDI*15] FISCHER, PHILIPP, DOSOVITSKIY, ALEXEY, ILG, EDDY, et al. "Flownet: Learning optical flow with convolutional networks". *arXiv preprint arXiv:1504.06852* (2015) 3.

[GFL*21] GUO, JIE, FU, XIHAO, LIN, LIQIANG, et al. "ExtraNet: real-time extrapolated rendering for low-latency temporal supersampling". *ACM Transactions on Graphics (TOG)* 40.6 (2021), 1–16 3.

[HTL18] HUI, TAK-WAI, TANG, XIAOOU, and LOY, CHEN CHANGE. "Liteflownet: A lightweight convolutional neural network for optical flow estimation". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, 8981–8989 3, 7.

[HZH*22] HUANG, ZHEWEI, ZHANG, TIANYUAN, HENG, WEN, et al. "Real-time intermediate flow estimation for video frame interpolation". *European Conference on Computer Vision*. Springer. 2022, 624–642 2.

[JSJ*18] JIANG, HUAIZU, SUN, DEQING, JAMPANI, VARUN, et al. "Super slomo: High quality estimation of multiple intermediate frames for video interpolation". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, 9000–9008 2, 3, 5.

[KJL*22] KONG, LINGTONG, JIANG, BOYUAN, LUO, DONGHAO, et al. "Ifrnet: Intermediate feature refine network for efficient frame interpolation". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, 1969–1978 1–3, 5, 6, 8.

[KSY21] KONG, LINGTONG, SHEN, CHUNHUA, and YANG, JIE. "Fastflownet: A lightweight network for fast optical flow estimation". *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, 10310–10316 3.

[LH17] LOSHCHILOV, ILYA and HUTTER, FRANK. "Decoupled weight decay regularization". *arXiv preprint arXiv:1711.05101* (2017) 6.

[Liu20] LIU, EDWARD. "Dlss 2.0-image reconstruction for real-time rendering with deep learning". *GPU Technology Conference (GTC)*. Vol. 1. 2. 2020 2–4.

[LWL*22] LU, LIYING, WU, RUIZHENG, LIN, HUAIJIA, et al. "Video frame interpolation with transformer". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, 3532–3542 2.

[LYT*17] LIU, ZIWEI, YEH, RAYMOND A, TANG, XIAOOU, et al. "Video frame synthesis using deep voxel flow". *Proceedings of the IEEE international conference on computer vision*. 2017, 4463–4471 3.

[LZH*23] LI, ZHEN, ZHU, ZUO-LIANG, HAN, LING-HAO, et al. "Amt: All-pairs multi-field transforms for efficient frame interpolation". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, 9801–9810 2, 3, 6.

[MES*23] MERCIER, ANTOINE, ERASMUS, RUAN, SAVANI, YASHESH, et al. "Efficient Neural Supersampling on a Novel Gaming Dataset". *Proceedings of the IEEE/CVF International Conference on Computer Vision. ICCV'23*. 2023 4.

[MHR18] MEISTER, SIMON, HUR, JUNHWA, and ROTH, STEFAN. "Unflow: Unsupervised learning of optical flow with a bidirectional census loss". *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018 5.

[NL18] NIKLAUS, SIMON and LIU, FENG. "Context-aware synthesis for video frame interpolation". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, 1701–1710 3.

[NL20] NIKLAUS, SIMON and LIU, FENG. "Softmax splatting for video frame interpolation". *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, 5437–5446 3.

[PKLK20] PARK, JUNHEUM, KO, KEUNSOO, LEE, CHUL, and KIM, CHANG-SU. "Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation". *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*. Springer. 2020, 109–125 3.

[RB17] RANJAN, ANURAG and BLACK, MICHAEL J. "Optical flow estimation using a spatial pyramid network". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, 4161–4170 3.

[RKT*22] REDA, FITSUM, KONTKANEN, JANNE, TABELLION, ERIC, et al. "Film: Frame interpolation for large motion". *European Conference on Computer Vision*. Springer. 2022, 250–266 2, 3, 6.

[SGHS98] SHADE, JONATHAN, GORTLER, STEVEN, HE, LI-WEI, and SZELISKI, RICHARD. "Layered depth images". *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 1998, 231–242 4.

[SYLK18] SUN, DEQING, YANG, XIAODONG, LIU, MING-YU, and KAUTZ, JAN. "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, 8934–8943 3.

[TD20] TEED, ZACHARY and DENG, JIA. "Raft: Recurrent all-pairs field transforms for optical flow". *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer. 2020, 402–419 3.

[WKZ*23] WU, SONGYIN, KIM, SUNGYE, ZENG, ZHENG, et al. "ExtraSS: A Framework for Joint Spatial Super Sampling and Frame Extrapolation". *SIGGRAPH Asia 2023 Conference Papers*. 2023, 1–11 3.

[WZH*23] WU, ZHIZHEN, ZUO, CHENYU, HUO, YUCHI, et al. "Adaptive Recurrent Frame Prediction with Learnable Motion Vectors". *SIGGRAPH Asia 2023 Conference Papers*. 2023, 1–11 2, 3.

[XCW*19] XUE, TIANFAN, CHEN, BAIAN, WU, JIAJUN, et al. "Video enhancement with task-oriented flow". *International Journal of Computer Vision* 127 (2019), 1106–1125 3.

[XNC*20] XIAO, LEI, NOURI, SALAH, CHAPMAN, MATT, et al. "Neural supersampling for real-time rendering". *ACM Transactions on Graphics (TOG)* 39.4 (2020), 142–1 2, 4.

[XSS*19] XU, XIANGYU, SIYAO, LI, SUN, WENXIU, et al. "Quadratic video interpolation". *Advances in Neural Information Processing Systems* 32 (2019) 3.

[YLS20] YANG, LEI, LIU, SHIQIU, and SALVI, MARCO. "A survey of temporal antialiasing techniques". *Computer graphics forum*. Vol. 39. 2. Wiley Online Library. 2020, 607–621 3.

[YZZ*24] YANG, SIPENG, ZHU, QINGCHUAN, ZHUGE, JUNHAO, et al. "Mob-FGSR: Frame Generation and Super Resolution for Mobile Real-Time Rendering". *ACM SIGGRAPH 2024 Conference Papers*. 2024, 1–11 1–4, 6.

[ZLY*21] ZENG, ZHENG, LIU, SHIQIU, YANG, JINGLEI, et al. "Temporally Reliable Motion Vectors for Real-time Ray Tracing". *Computer Graphics Forum*. Vol. 40. 2. Wiley Online Library. 2021, 79–90 3.

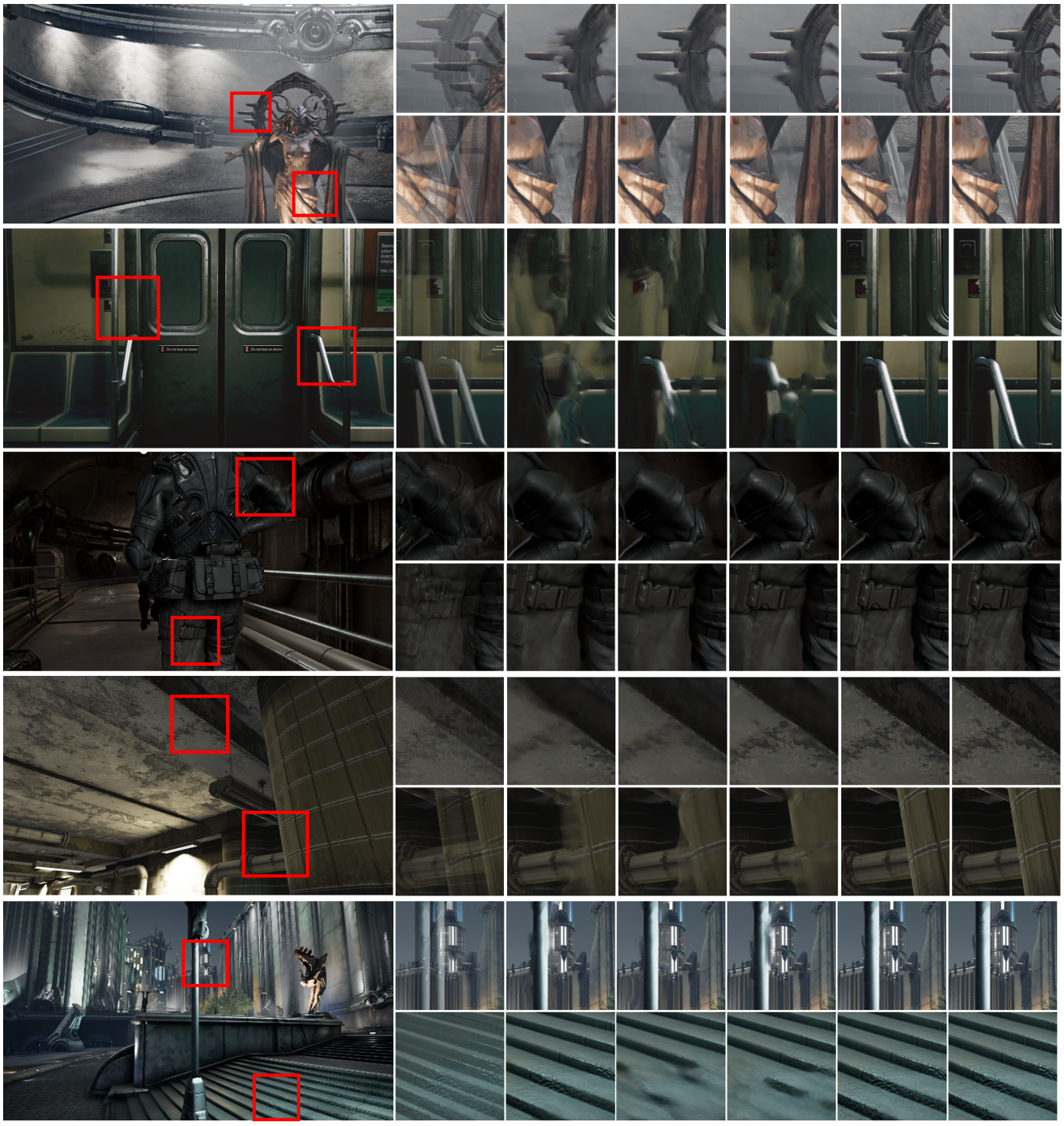


Figure 9: *Qualitative comparison of our method against state-of-the-art deep learning methods.*

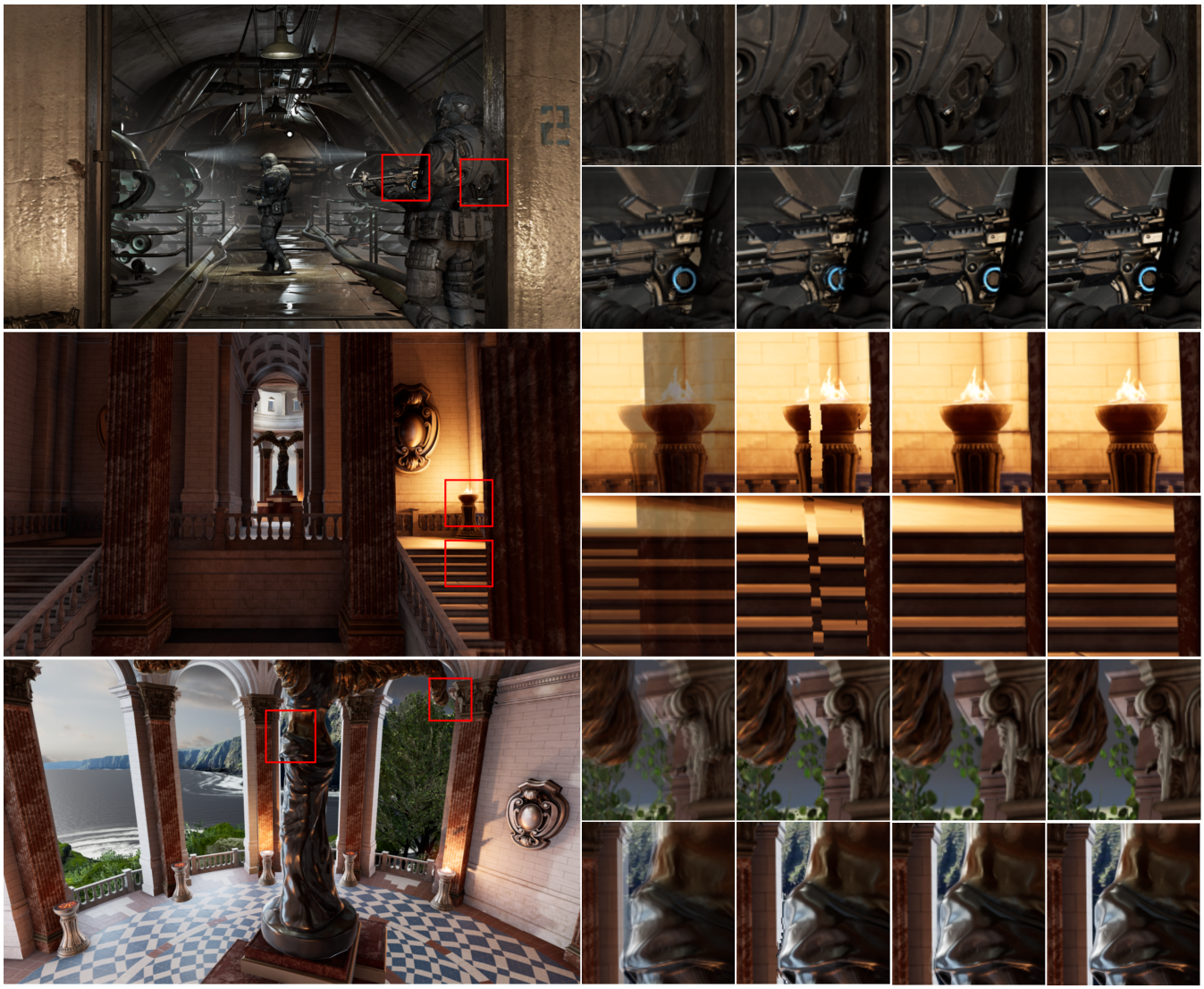


Figure 10: *Qualitative comparison of our method against Mob-FGSR.*