ALIGN-FG: EFFICIENT GAME FRAME GENERATION WITH TARGET-ALIGNED MOTION AND PROGRESSIVE REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Game frame generation improves temporal smoothness by synthesizing intermediate frames with limited additional rendering cost. Existing two-stage methods generate complementary warped candidates using image-based optical flow and engine-provided motion vectors (MVs), and subsequently fuse them for frame synthesis. However, intermediate motion derived from engine MVs under a linear motion assumption can misalign the estimated initial positions with their targets, while standalone optical-flow estimation introduces substantial computational overhead. We present Align-FG, an efficient framework that addresses both limitations with two complementary designs. First, Target-Aligned Motion Generation (TAMG) uses engine MVs and depth during training to construct a target-aligned bidirectional geometric prior, reducing initialization errors and allowing the network to focus on appearance changes that are poorly described by engine MVs, such as reflections, shadows, and semi-transparent effects. At inference, the same motion representation is constructed from endpoint MVs under a linear motion assumption, enabling deployment without access to the target frame. Second, we develop a lightweight one-stage Progressive Motion Fusion and Refinement (PMFR) network that adaptively fuses propagated flow with the geometric motion prior and predicts residual corrections across scales, incorporating appearance cues without a standalone optical-flow estimator. Experiments on our Unreal Engine-based and real-game datasets show that Align-FG improves perceptual quality over the evaluated baselines, while generating a 1280×720 frame in 1.60 ms on an NVIDIA L20 GPU, approximately 6–$7\times$ faster than RIFE and IFRNet.

1 INTRODUCTION

High-frame-rate rendering is fundamental to delivering smooth, responsive, and immersive gaming experiences. However, rendering every frame at high frame rates incurs substantial computational and power costs, making it particularly challenging for resource-constrained edge devices, such as mobile phones and tablets. Game frame generation provides an efficient alternative by synthesizing intermediate frames between rendered frames, improving temporal smoothness without proportionally increasing the rendering workload.

A straightforward solution is to treat rendered game frames as a video sequence and apply video frame interpolation (VFI). Most efficient VFI methods rely on estimating optical flow to establish pixel correspondences between adjacent frames and synthesize intermediate content through warping and refinement. While such methods can achieve real-time performance, their motion estimation can become unreliable under large displacement, rapid camera rotation, occlusion, and complex scene dynamics, often leading to suboptimal interpolation quality and noticeable visual artifacts.

Unlike generic VFI, engine-assisted game frame generation can use auxiliary information from the rendering pipeline, particularly depth and motion vectors (MVs), which can be obtained with minimal computational overhead. Depth provides geometric cues for visibility and occlusion reasoning, while engine MVs encode pixel-wise displacement from the current frame to the previous frame. Traditional methods, such as Mob-FG (Yang et al., 2024), mainly rely on engine-provided MVs to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

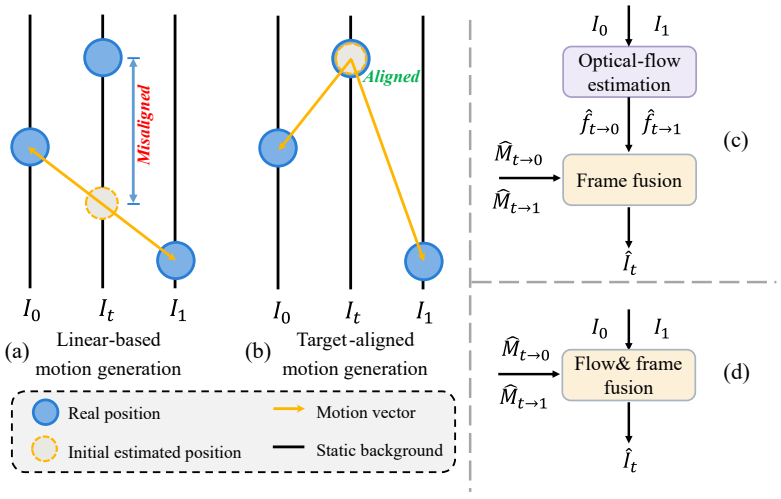


Figure 1: Motivation and design of Align-FG. **(a–b) Motion generation:** linear-based generation can misalign intermediate positions under nonlinear motion, whereas TAMG provides target-aligned priors during training. **(c–d) Network architecture:** a two-stage pipeline separates optical-flow estimation and frame fusion, whereas our single-stage model unifies motion fusion and frame synthesis.

reconstruct intermediate frames under simplified motion assumptions. While these MVs are effective for regions with explicit geometric motion, they may not reliably capture appearance changes caused by shadows, lighting, particles, reflections, and semi-transparent effects. As a result, relying solely on engine motion often causes inaccurate alignment and temporal flickering in these regions.

To address this limitation, subsequent methods introduce image-estimated optical flow to complement engine MVs. For example, FSR 3 (Advanced Micro Devices, 2023) warps neighboring frames using both engine MVs and optical flow, and fuses the resulting warped frames using hand-crafted rules based on image similarity. However, such rules can be unreliable when image similarity does not accurately indicate motion correspondence. NFRU (Arm, 2026) replaces these hand-crafted rules with a learned network that predicts fusion weights for frames warped using engine MVs and conventional optical flow. PC-oriented systems, including DLSS 4 (Corporation, 2025), XeSS 2 (Intel Corporation, 2024), and FSR Frame Generation (Advanced Micro Devices, 2026), further incorporate learning-based optical-flow estimation to improve frame quality, albeit at a higher computational cost.

Despite these advances, we identify two practical challenges in efficient game frame generation:

First, intermediate motion derived from engine MVs typically relies on a linear motion assumption, which can misalign the estimated intermediate positions with their targets under nonlinear motion, as illustrated in Figure 1(a). Such misalignment is subsequently propagated through the warping and fusion stages, increasing the burden on the network to correct motion errors while handling occlusions, disocclusions, and appearance changes.

Second, balancing interpolation quality and computational efficiency remains challenging. Figure 1(c) illustrates a two-stage design: optical flow is estimated separately and then combined with engine MVs in a frame-fusion stage. The final quality and runtime therefore depend on both stages. Lightweight approaches use conventional optical-flow methods, such as block matching (De Haan & Biezen, 1994) or 3DRS (De Haan et al., 1993), together with learned fusion networks to reduce computational cost, but their performance is often limited by inaccurate flow estimates in challenging regions. In contrast, learning-based optical-flow estimation (Dosovitskiy et al., 2015; Teed & Deng, 2020) can provide more reliable image-derived motion cues and improve interpolation quality, but at substantially higher computational and power costs. This quality–efficiency trade-off is particularly critical for resource-constrained devices.

In this paper, we address these challenges with two complementary designs. To mitigate the target-position misalignment introduced by linear motion assumptions, we propose Target-Aligned Motion Generation (TAMG). During training, TAMG leverages engine MVs and depth to project motion cues into the target-frame coordinate space and construct a target-aligned bidirectional motion pair.

As illustrated in Figure 1(a–b), TAMG aligns valid geometric regions with their target positions, in contrast to the spatial misalignment caused by linear motion initialization. The resulting geometric motion prior allows the network to focus on regions where engine motion is unreliable. At inference, the initial bidirectional motion pair is constructed from the available endpoint MVs under a linear motion assumption, maintaining a consistent motion representation between training and inference.

To address the quality–efficiency trade-off, we further propose Progressive Motion Fusion and Refinement (PMFR), a lightweight single-stage interpolation network with a coarse-to-fine decoder. As illustrated in Figure 1(c–d), PMFR replaces the conventional separate optical-flow estimation and frame-fusion pipeline with unified progressive motion reconstruction. At each decoder stage, a refinement block adaptively fuses the propagated flow with the current-scale geometric motion prior and predicts residual motion corrections. This coarse-to-fine fusion-and-refinement process progressively improves motion accuracy while directly supporting frame synthesis, without relying on a standalone optical-flow estimator.

Our contributions are summarized as follows:

- We propose Target-Aligned Motion Generation (TAMG), which constructs a target-aligned bidirectional motion pair during training to mitigate the spatial misalignment introduced by linear motion assumptions. TAMG aligns reliable geometric motion with the target positions while maintaining a consistent motion representation between training and inference.
- We develop PMFR, a lightweight multi-scale interpolation network that progressively fuses and refines complementary motion cues across decoder stages. PMFR improves motion accuracy and frame interpolation quality without relying on a standalone learning-based optical-flow estimator.
- We evaluate our method on both Unreal Engine-based and real-game datasets, demonstrating improved perceptual quality over the evaluated baselines. Measurements on an NVIDIA L20 GPU show approximately 6–7$\times$ lower latency than RIFE and IFRNet, and comparable network FLOPs to NFRU.

2 Related Work

2.1 Video Frame Interpolation

Video frame interpolation synthesizes intermediate frames from neighboring inputs. Early methods rely on optical-flow estimation and motion-compensated warping (Jiang et al., 2018), with depth later introduced for occlusion reasoning (Bao et al., 2019). Efficient approaches directly estimate intermediate motion, such as RIFE with privileged distillation (Huang et al., 2022) and IFRNet with joint feature-motion refinement (Kong et al., 2022). Subsequent works improve motion modeling through scale-agnostic estimation (Reda et al., 2022), all-pairs matching (Li et al., 2023), Transformer-based designs (Lu et al., 2022; Zhang et al., 2023), and asymmetric bilateral motion estimation (Park et al., 2021). Diffusion-based methods further improve modeling capacity for ambiguous motion (Danier et al., 2024; Zhang et al., 2025; 2026), but incur substantially higher computation. Despite these advances, image-based VFI remains challenged by large motion, rapid camera rotation, disocclusion, and complex rendered effects in games, while also lacking access to rendering-pipeline geometry. In contrast, our method exploits engine-provided MVs and depth for target-aligned motion initialization and efficient progressive motion fusion and refinement.

2.2 Game Frame Generation

Engine-motion-based frame generation. Game frame generation can exploit auxiliary rendering signals, particularly depth and engine motion vectors (MVs), which provide explicit geometric information for efficient frame reconstruction. Early methods primarily rely on these signals to warp rendered frames without introducing a separate optical-flow estimator. Mob-FG (Yang et al., 2024), for example, derives intermediate motion from engine MVs under simplified motion assumptions. Such approaches are highly efficient for geometry-driven regions, but engine MVs provide limited guidance for appearance changes caused by shadows, illumination, particles, reflections, and semi-transparent effects, often resulting in misalignment and temporal artifacts.

Heuristic motion fusion. To complement engine motion, subsequent methods incorporate image-based optical flow and construct multiple warped candidates from different motion sources. FSR 3 (Advanced Micro Devices, 2023) and related lightweight approaches (Yang et al., 2025) fuse these candidates using hand-crafted cues, such as image similarity or consistency measures. Similar frame-generation techniques have also been deployed on mobile platforms, e.g., iQOO 13 (iQOO, 2024). While inexpensive, heuristic fusion can become unreliable in complex rendered regions, where appearance similarity may not faithfully indicate the correct motion correspondence.

Learning-based fusion and synthesis. Recent methods replace hand-crafted fusion with learned reconstruction. NFRU (Arm, 2026) combines conventional optical flow with a learned fusion network to adaptively blend candidates generated from image-based and engine-provided motion. Its performance, however, remains constrained by the quality of conventional optical flow, particularly under large motion and complex visual effects. More powerful PC-oriented systems, including DLSS 4 (Corporation, 2025), XeSS 2 (Intel Corporation, 2024), and FSR Frame Generation (Advanced Micro Devices, 2026), further employ learned components for motion estimation and/or frame synthesis, achieving higher reconstruction quality with larger computational budgets. In contrast, Align-FG combines target-aligned motion initialization with a lightweight Progressive Motion Fusion and Refinement (PMFR) network, which progressively fuses and refines complementary motion cues in a coarse-to-fine decoder without relying on a standalone optical-flow estimator.

3 METHOD

3.1 OVERVIEW

We aim to efficiently synthesize an intermediate game frame from two rendered endpoint frames. Each training sample contains three recorded frames $I_0, I_t, I_1 \in \mathbb{R}^{H\times W\times 3}$, the depth map D_1, and the engine's backward MVs $M_{t\to 0}$ and $M_{1\to t}$. The endpoint times are normalized to 0 and 1, and $t \in (0, 1)$ is the target time. We use the convention that $M_{a\to b}(\mathbf{p})$ is defined at pixel $\mathbf{p}$ in frame a and points to $\mathbf{p} + M_{a\to b}(\mathbf{p})$ in frame b. All motion vectors are expressed in pixel units on the grid where they are used. During inference, only I_0, I_1, D_1, and the full-interval backward MV $M_{1\to 0}$ are required to synthesize the intermediate frame.

Align-FG uses Target-Aligned Motion Generation (TAMG) during training and linear-based motion generation during inference (Section 3.2). Both schemes construct engine-based geometric motion priors that provide the initial bidirectional motion for the Progressive Motion Fusion and Refinement (PMFR) network (Section 3.3). Given the endpoint frames and this initial motion, the single-stage PMFR network progressively fuses and refines motion in a coarse-to-fine decoder to synthesize the intermediate frame.

3.2 GEOMETRIC MOTION PRIOR GENERATION

We construct a bidirectional geometric motion prior on the target-frame grid to provide the PMFR network with an initial motion estimate. During training, TAMG uses recorded target-frame correspondences rather than estimating intermediate positions through linear scaling (Figure 2(a)). At inference, linear-based motion generation constructs the prior from the full-interval endpoint MV (Figure 2(b)). Both schemes use depth-aware motion splatting (DMS) to project motion onto the target grid, resolve projection collisions using depth, and identify locations without projected correspondences.

Depth-aware motion splatting. Given a source pixel $\mathbf{p}$ with backward MV $M(\mathbf{p})$ and depth $D(\mathbf{p})$, DMS projects its motion onto the target grid. Let $T \in [0, 1]$ denote the normalized target time within the interval spanned by this MV, with $T = 0$ at the earlier endpoint and $T = 1$ at the source frame. To reduce small holes caused by discretization, we scatter the source vector to a four-pixel footprint:

$$\begin{aligned} \mathbf{s}_\mathbf{p} &= \mathbf{p} + (1 - T)M(\mathbf{p}), \\ \mathcal{N}_4(\mathbf{s}_\mathbf{p}) &= \{\lfloor \mathbf{s}_\mathbf{p} \rfloor + (i, j) : i, j \in \{0, 1\}\}. \end{aligned} \tag{1}$$

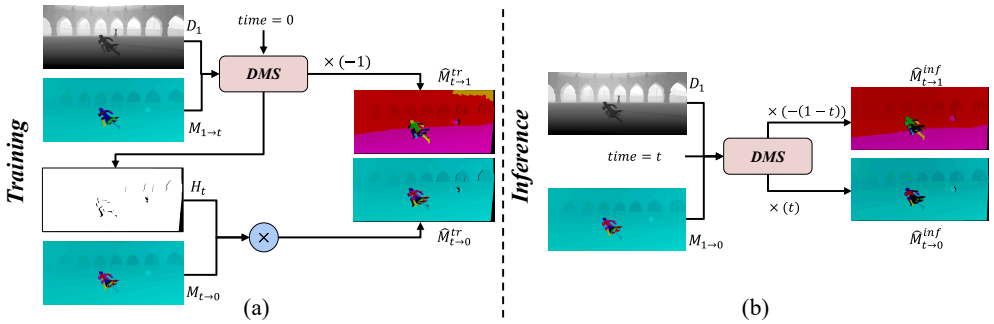


Figure 2: Geometric motion prior generation. (a) During training, TAMG projects the recorded target-frame correspondences onto the target grid. (b) During inference, linear-based motion generation rescales the full-interval endpoint MV to the intermediate time step. Both schemes apply depth-aware motion splatting (DMS) to produce the bidirectional motion prior and its coverage mask.

Here $\lfloor \cdot \rfloor$ denotes coordinate-wise rounding down. Each in-bounds pixel in $\mathcal{N}_4(\mathbf{s_p})$ receives the same vector $M(\mathbf{p})$. When multiple source pixels cover the same target pixel, we retain the vector associated with the smallest source depth, favoring the surface closest to the camera.

The operator returns $(\bar{M}, H) = \mathrm{DMS}_T(M, D)$, where $\bar{M}$ stores the selected source vectors on the target grid and H is a binary coverage mask. We set $H(\mathbf{q}) = 1$ if target pixel $\mathbf{q}$ receives at least one projected vector, and $H(\mathbf{q}) = 0$ otherwise. Uncovered entries of $\bar{M}$ are set to zero.

Target-aligned motion generation for training. During training, TAMG constructs a bidirectional motion prior from recorded target-frame correspondences (Figure 2(a)). The engine MV $M_{1\to t}$ maps pixels from the current frame I_1 to the recorded intermediate frame I_t. Within this MV interval $[t, 1]$, the local parameter $T = 0$ corresponds to the target time t. Applying DMS with this setting projects the full displacement onto the target grid, yielding

$$\begin{aligned} (\bar{M}^{\mathrm{tr}}, H_t^{\mathrm{tr}}) &= \mathrm{DMS}_0(M_{1\to t}, D_1), \\ \hat{M}^{\mathrm{tr}}_{t\to0} = H_t^{\mathrm{tr}} \odot M_{t\to0}, &\qquad \hat{M}^{\mathrm{tr}}_{t\to1} = -\bar{M}^{\mathrm{tr}}. \end{aligned} \tag{2}$$

Negating the splatted MV gives a discrete approximation of the target-to-current motion. The target-to-previous prior is obtained from the recorded MV $M_{t\to0}$ by applying the coverage mask H_t^{tr}, where $\odot$ denotes element-wise multiplication with the mask broadcast over motion channels.

Applying the same mask gives both training priors the same projected support, consistent with the missing-motion representation used at inference. An uncovered target location has no projected correspondence from I_1, but its content may still be visible in I_0.

Linear-based motion generation for inference. At inference, the intermediate frame and its MVs are unavailable. We therefore construct the prior from the full-interval backward MV $M_{1\to0}$ (Figure 2(b)). Under linear motion, a pixel $\mathbf{p}$ in I_1 reaches $\mathbf{p} + (1-t)M_{1\to0}(\mathbf{p})$ at time t. Since this MV spans the interval $[0, 1]$, setting the local DMS parameter to $T = t$ yields

$$\begin{aligned} (\bar{M}^{\mathrm{inf}}, H_t^{\mathrm{inf}}) &= \mathrm{DMS}_t(M_{1\to0}, D_1), \\ \hat{M}^{\mathrm{inf}}_{t\to0} = t\bar{M}^{\mathrm{inf}}, &\qquad \hat{M}^{\mathrm{inf}}_{t\to1} = -(1-t)\bar{M}^{\mathrm{inf}}. \end{aligned} \tag{3}$$

The factors t and $-(1-t)$ convert the projected endpoint MV into target-to-previous and target-to-current motion priors, respectively. Both fields share the projected coverage H_t^{inf}.

Motion prior initialization. In both training and inference, H_t marks target locations covered by projections from I_1. Uncovered locations can arise from occlusion, sampling gaps, or motion beyond the image boundaries. Thus, H_t represents projection coverage rather than an exact visibility mask.

Omitting the phase superscripts for brevity, we concatenate the two motion directions as $\hat{M}_{\mathrm{eng}} = [\hat{M}_{t\to0}, \hat{M}_{t\to1}]$. This prior, together with the endpoint frames, is passed to the PMFR network (Section 3.3) for motion refinement and frame synthesis.

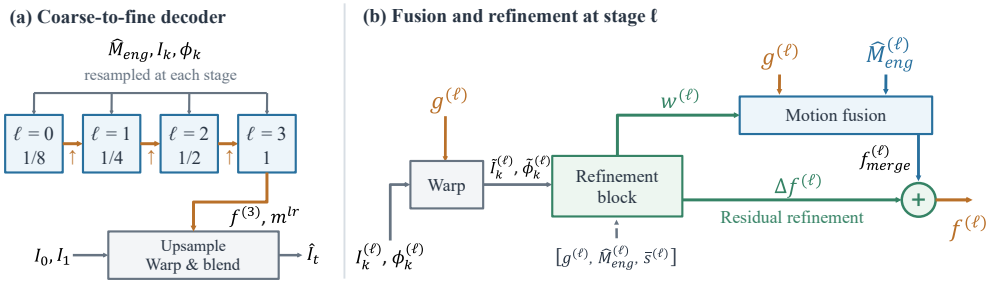


Figure 3: Architecture of the PMFR network. (a) Four-stage coarse-to-fine motion refinement followed by full-resolution synthesis. (b) Each finer-scale refinement block predicts the fusion gate and residual motion update in Eq. (5), using endpoint features $\phi_k^{(\ell)}$, warped tensors denoted by tildes, and the aggregated state $\bar{s}^{(\ell)}$ containing upsampled features and fusion logits. Auxiliary branches are omitted for clarity.

3.3 Progressive Motion Fusion and Refinement Network

PMFR is our lightweight interpolation network, comprising a shared feature encoder and a coarse-to-fine decoder with four refinement blocks. Each block adaptively fuses the geometric motion prior with its current motion estimate and predicts a residual correction. Applying these blocks across scales progressively refines motion for frame synthesis.

Architecture. As shown in Figure 3(a), a shared convolutional head encodes the endpoint frames, and four refinement stages operate at scales $1/8, 1/4, 1/2, 1$ relative to the network input. We denote the prepared geometric prior at stage ℓ by $\hat{M}_{\text{eng}}^{(\ell)}$, expressed in that stage's pixel units. Each refinement block jointly predicts a flow residual, a per-pixel fusion gate, and an image-blending weight. Layer specifications are given in Appendix A.

Progressive fusion and refinement. Let $f^{(\ell)} = [f_{t\to0}^{(\ell)}, f_{t\to1}^{(\ell)}]$ denote the bidirectional flow at stage $\ell \in \{0,1,2,3\}$. The coarsest stage takes the unwarped endpoint images, their features, and the geometric prior as input. We set $g^{(0)} = \mathbf{0}$ for a unified formulation and propagate flow to finer stages as

$$g^{(\ell)} = \text{UP}_{2\times}(f^{(\ell-1)}), \qquad \ell = 1,2,3, \tag{4}$$

where $\text{UP}_{2\times}$ bilinearly upsamples the flow and doubles its values to match the new pixel units. At each finer stage (Figure 3(b)), $g^{(\ell)}$ backward-warps both endpoint images and their features to the target grid. The refinement block takes these warped observations, the two motion candidates $[g^{(\ell)}, \hat{M}_{\text{eng}}^{(\ell)}]$, and the upsampled feature state and raw gate/blending logits from the preceding stage. Each refinement block jointly predicts a per-pixel gate $w^{(\ell)} \in [0,1]$ and residual $\Delta f^{(\ell)}$, yielding

$$\begin{aligned} f_{\text{merged}}^{(\ell)} &= w^{(\ell)} \odot g^{(\ell)} + (1 - w^{(\ell)}) \odot \hat{M}_{\text{eng}}^{(\ell)}, \\ f^{(\ell)} &= f_{\text{merged}}^{(\ell)} + \Delta f^{(\ell)}, \qquad \ell = 0,1,2,3. \end{aligned} \tag{5}$$

The gate is shared across the four flow channels. At finer stages, larger values favor the propagated estimate, while smaller values favor the geometric prior; the residual corrects the fused motion using appearance cues. With $g^{(0)} = \mathbf{0}$, the same update initializes flow from a gated prior and a residual. The corresponding gate parameterization is given in Appendix A.

Low-resolution prediction and full-resolution synthesis. PMFR operates at reduced resolution and performs final synthesis at full resolution. By default, the network predicts target-to-endpoint flows and a blending mask m^{lr} at $H/2 \times W/2$. Let $\mathcal{B}_{2\times}$ denote bilinear upsampling without value scaling. The full-resolution predictions are

$$f_{t\to k} = 2\mathcal{B}_{2\times}\left(f_{t\to k}^{(3)}\right), \quad k \in \{0,1\}, \qquad m = \mathcal{B}_{2\times}(m^{\text{lr}}),$$

where only the flow is scaled to convert displacements to full-resolution pixel units. The output frame is synthesized from the original endpoint images as

$$\hat{I}_t = m \odot \mathrm{warp}(I_0, f_{t\to 0}) + (1 - m) \odot \mathrm{warp}(I_1, f_{t\to 1}). \tag{6}$$

Here, $\mathrm{warp}(I, f)(\mathbf{q}) = I(\mathbf{q} + f(\mathbf{q}))$ denotes backward warping with bilinear sampling. This design reduces network computation while preserving full-resolution content for synthesis.

4 Experiments

4.1 Experimental Setup

Datasets. We use two datasets, UE Demo and Real Game, recorded at 60 FPS with per-frame color, depth, and backward motion-vector buffers. UE Demo is captured at 1280×720 and includes four maps: FantasticVillage (F.Village), AsianVillage (A.Village), StylizedEgypt (Egypt), and Highrise. Each map contains three representative gameplay scenarios: routine traversal, combat with visual effects, and large-angle camera rotation, with 3,000 training and 600 validation frames per map. Real Game is captured at 1800×810 and covers the same three scenario types in more complex real-game environments. We categorize them as Easy, Medium, and Hard, with 3,000, 3,000, and 5,000 training frames and 600 validation frames per category, respectively. For computational efficiency, all methods are trained and evaluated at half resolution.

Preprocessing and training. Engine MVs are downsampled to half resolution with depth-guided block selection, with vector values scaled to the corresponding pixel units. We optimize $\mathcal{L} = \lambda_{\mathrm{rec}}\mathcal{L}_1 + \lambda_{\mathrm{vgg}}\mathcal{L}_{\mathrm{vgg}}$, where $\mathcal{L}_1$ measures the pixelwise L_1 difference between $\hat{I}_t$ and I_t, and $\mathcal{L}_{\mathrm{vgg}}$ compares their VGG features for perceptual similarity. Additional implementation details are provided in Appendix A.

Baselines and metrics. We compare against RGB-only interpolators RIFE (Huang et al., 2022) and IFRNet (Kong et al., 2022), engine-assisted methods Mob-FG (Yang et al., 2024), and NFRU (Arm, 2026). All baselines are reproduced using the original authors' publicly released code. To ensure a fair comparison, all methods use the same source data, training and validation splits, and data augmentation pipeline. We report PSNR and SSIM (Wang et al., 2004) for reconstruction fidelity, LPIPS with the AlexNet backbone for perceptual similarity, and FloLPIPS (Danier et al., 2022) for video perceptual quality.

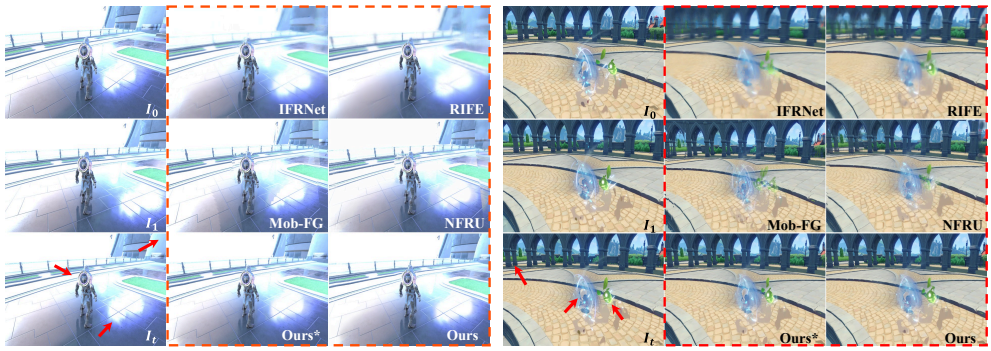
4.2 Comparison with State-of-the-Art Methods

Reconstruction and perceptual quality. Table 1 shows that Ours, evaluated with the linear-based prior, achieves the lowest LPIPS in all seven groups and the lowest FloLPIPS in six. Its average LPIPS and FloLPIPS are 4.46 and 7.24, respectively, improving on NFRU by 34.8% and 30.7%. Although its average PSNR and SSIM remain below those of RIFE and IFRNet, Ours leads both metrics on F.Village and Real Game Hard. These results highlight its perceptual advantage, particularly under challenging motion. Ours* achieves further gains using TAMG at evaluation; this target-assisted configuration is reported as a privileged reference rather than a deployable method.

Qualitative comparison. Figure 4 compares results on (a) UE Demo and (b) Real Game. In (a), floor illumination changes with camera motion. Like shadows and semi-transparent effects, these changes are not fully captured by engine MVs and require image-based motion cues. IFRNet and RIFE capture the illumination displacement but distort the background under large motion. Mob-FG and NFRU reproduce the illumination less faithfully and show artifacts near character boundaries and disoccluded regions with incomplete geometric correspondences. Both Ours* and Ours better preserve background structure and reproduce illumination closer to the ground truth. Similar improvements in visual effects and background details are observed in (b). Additional qualitative results are provided in Appendix Figure 6. More inference-time video comparisons with other methods, including DLSS 4, are provided in the supplementary material.

Table 1: Comparison on four UE Demo scenes and three Real Game groups. Ours* uses the privileged TAMG prior at evaluation and is shown only as a non-deployable reference. Yellow and blue mark the best and second-best deployable results, respectively, including ties.

		UE Demo				Real Game			
Metric	Method	F.Village	A.Village	Egypt	Highrise	Easy	Medium	Hard	Average
LPIPS(%)↓	RIFE	16.59	11.63	10.62	9.07	5.46	7.52	25.30	12.31
	IFRNet	14.22	10.51	8.79	7.54	5.20	7.01	21.70	10.71
	Mob-FG	7.00	5.21	4.63	4.02	4.14	5.30	19.31	7.09
	NFRU	7.21	5.41	4.63	4.19	3.77	5.88	16.81	6.84
	Ours	4.90	4.78	2.55	3.13	3.46	5.01	7.38	4.46
	Ours*	2.50	2.48	1.29	1.60	2.35	3.40	4.94	2.65
FloLPIPS(%)↓	RIFE	21.83	16.45	14.29	11.04	9.42	16.00	29.50	16.93
	IFRNet	18.45	14.42	11.56	9.18	8.34	13.78	25.37	14.44
	Mob-FG	10.64	8.43	8.07	5.76	8.55	12.86	27.68	11.71
	NFRU	10.76	8.42	7.84	6.15	6.20	10.96	22.85	10.45
	Ours	7.71	6.80	4.79	3.42	6.22	10.33	11.38	7.24
	Ours*	6.15	5.48	3.84	2.74	5.61	9.00	9.83	6.09
PSNR(dB)↑	RIFE	26.48	29.96	32.60	28.89	30.04	30.14	24.31	28.92
	IFRNet	26.38	29.88	32.67	28.92	29.68	29.71	23.90	28.73
	Mob-FG	25.35	28.85	30.98	27.74	27.38	27.90	21.41	27.09
	NFRU	25.43	29.04	31.27	28.37	28.40	27.28	21.84	27.38
	Ours	27.48	29.12	31.48	28.53	28.58	28.39	25.77	28.48
	Ours*	32.80	34.79	37.73	34.20	31.28	31.06	28.54	32.91
SSIM(%)↑	RIFE	81.15	85.67	92.40	90.61	92.22	92.22	77.90	87.45
	IFRNet	81.20	85.72	92.46	90.82	92.10	91.19	77.28	87.25
	Mob-FG	78.95	84.78	91.32	89.20	90.19	90.34	71.21	85.14
	NFRU	78.57	84.87	91.64	89.59	91.12	90.15	71.52	85.35
	Ours	85.89	85.54	88.83	87.49	90.08	89.03	78.65	86.50
	Ours*	95.10	94.76	98.06	96.87	94.89	93.99	90.97	94.95



(a) UE demo (b) Real game

Figure 4: Qualitative comparisons on (a) UE Demo and (b) Real Game. The interpolation results of all methods are shown within the red dashed box. Ours* uses TAMG at evaluation, whereas Ours uses the linear-based prior. Red arrows mark challenging lighting, effect, and boundary regions.

Efficiency analysis. At 1280×720 on an NVIDIA L20 (Table 2), Align-FG runs in 1.60 ms, translating reduced computation into $7.1\times/6.0\times$ speedups over RIFE/IFRNet under the same backend. Compared with NFRU, it improves perceptual quality at similar network FLOPs, with 0.30 ms additional latency. Its higher postprocessing cost highlights overhead beyond network computation. Mob-FG is faster but yields lower perceptual quality, illustrating a quality–latency trade-off. Different backends limit direct comparisons with NFRU and Mob-FG.

4.3 Ablation Studies

We ablate the two key designs of Align-FG: the target-aligned training prior and the fusion and refinement components of the PMFR network. Unless a factor is explicitly varied, comparisons use the same network architecture, data split, optimization schedule, and linear inference prior, evaluated on the four UE Demo scenes.

Target-aligned training prior. Table 3 separates the effects of training priors and privileged motion at evaluation. Linear/Linear and our default TAMG/Linear configuration use the same linear evaluation prior, isolating the benefit of target-aligned training: PSNR improves from 28.77 to 29.15 dB and LPIPS decreases from 4.57 to 3.84. TAMG/TAMG uses recorded target-frame motion at evaluation and serves only as a privileged diagnostic. Its 35.01 dB PSNR and 1.94 LPIPS suggest further potential with more accurate priors, rather than a guaranteed deployable upper bound.

Table 2: Efficiency at 1280×720 on an NVIDIA L20 GPU. FLOPs count network computation only. Pre/Net/Post/Total denote preprocessing, network inference, postprocessing, and total latency. CG denotes CUDA Graph; all neural networks use FP32.

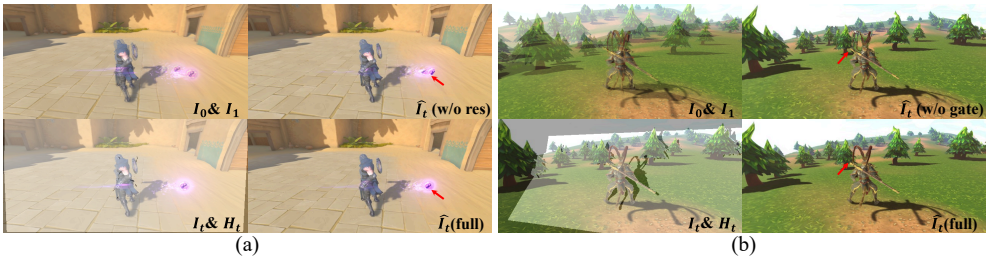
Method	Params (M)	FLOPs (G)	Latency (ms)				Backend
			Pre	Net	Post	Total	
RIFE	10.71	328.6	0.07	11.24	0.03	11.34	PyTorch + CG
IFRNet	2.80	244.0	0.01	9.59	0.03	9.63	PyTorch + CG
NFRU	0.10	5.65	0.15	1.10	0.05	1.30	Slang + CG
Mob-FG	–	–	–	–	–	0.57	GLSL (rasterization)
Ours	0.57	5.63	0.09	1.29	0.23	1.60	PyTorch + CG

Table 3: Effect of training and evaluation motion priors.

Train	Eval	LPIPS (%)	FloLPIPS (%)	PSNR (dB)	SSIM (%)
Linear	Linear	4.57	6.15	28.77	86.21
TAMG	TAMG	1.94	4.48	35.01	96.29
TAMG	Linear	3.84	5.68	29.15	86.94

Table 4: Effect of adaptive fusion and residual refinement in PMFR.

Gate	Res.	LPIPS (%)	FloLPIPS (%)	PSNR (dB)	SSIM (%)
✓	–	4.61	6.24	28.88	86.76
–	✓	3.98	5.95	28.88	86.75
✓	✓	3.84	5.68	29.15	86.94



(a) (b)

Figure 5: Qualitative ablations of the PMFR network. Left: removing residual updates (w/o res) degrades reconstruction of semi-transparent effects. Right: removing the learned gate (w/o gate) breaks fine structures under large camera rotation. The full model better preserves both regions, highlighted by red arrows.

Progressive motion fusion and refinement. Table 4 evaluates adaptive fusion and residual refinement under the same linear inference prior. Removing residual refinement increases LPIPS from 3.84 to 4.61; replacing the learned gate with fixed averaging increases it to 3.98. The full model performs best across all four metrics, supporting their complementary roles.

Figure 5 illustrates these effects. Without residual refinement, uncorrected motion degrades semi-transparent effects. Without the learned gate, thin structures such as the hair ribbon fragment under large camera rotation. The full model better preserves both by adaptively fusing motion cues and progressively correcting the resulting flow.

5 Conclusion

We presented Align-FG, a lightweight framework for engine-assisted game frame generation that combines target-aligned training with progressive motion fusion and refinement. TAMG constructs target-aligned geometric motion priors from recorded target-frame correspondences during training, while inference uses a deployable linear prior derived from endpoint motion. The single-stage PMFR network progressively fuses geometric priors with propagated motion estimates and predicts residual corrections across scales, incorporating appearance cues without a standalone optical-flow estimator. Experiments on UE Demo and Real Game demonstrate improved perceptual quality, particularly under large motion and complex visual effects, while ablation studies validate the benefits of target-aligned training and the complementary roles of adaptive fusion and residual refinement. Align-FG achieves 1.60 ms end-to-end latency at 1280×720 on an NVIDIA L20 GPU, approximately 6–7$\times$ faster than RIFE and IFRNet under the same backend. In future work, we plan to further quantize and deploy Align-FG on edge devices and evaluate its performance in real-game scenarios.

REFERENCES

Advanced Micro Devices. AMD FidelityFX Super Resolution 3: Integration overview. Version 3.0.3, revised December 14, 2023, 2023. URL https://gpuopen.com/presentations/2023/AMD_FidelityFX_Super_Resolution_3-Overview_and_Integration.pdf. Accessed September 23, 2026.

Advanced Micro Devices. AMD FSR Frame Generation 4.0.1. GPUOpen developer documentation, 2026. URL https://gpuopen.com/manuals/fsr_sdk/techniques/frame-interpolation-ml/. Version released June 24, 2026; accessed September 23, 2026.

Arm. Neural frame rate upscaling (NFRU). Official model card and technical resources, 2026. URL https://huggingface.co/Arm/neural-frame-rate-upscaling. Accessed September 23, 2026.

Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3703–3712, 2019.

NVIDIA Corporation. DLSS 4: Transforming Real-Time Graphics with AI. https://research.nvidia.com/labs/adlr/DLSS4/, 2025. Technical Report.

Duolikun Danier, Fan Zhang, and David Bull. FloLPIPS: A bespoke video quality metric for frame interpolation. In *2022 Picture Coding Symposium (PCS)*, pp. 283–287, 2022. doi: 10.1109/PCS56426.2022.10018062.

Duolikun Danier, Fan Zhang, and David Bull. LDMVFI: Video frame interpolation with latent diffusion models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(2):1472–1480, 2024. doi: 10.1609/aaai.v38i2.27912.

Gerard De Haan and Paul WAC Biezen. Sub-pixel motion estimation with 3-d recursive search block-matching. *Signal Processing: Image Communication*, 6(3):229–239, 1994.

Gerard De Haan, Paul WAC Biezen, Henk Huijgen, and Olukayode A Ojo. True-motion estimation with 3-d recursive search block matching. *IEEE transactions on circuits and systems for video technology*, 3(5):368–379, 1993.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766, 2015.

Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *Computer Vision – ECCV 2022*, pp. 624–642. Springer, 2022. doi: 10.1007/978-3-031-19781-9_36.

Intel Corporation. Intel Xe Super Sampling 2 whitepaper. Technical whitepaper, 2024. URL https://www.intel.com/content/www/us/en/developer/articles/technical/xess2-whitepaper.html. Accessed September 23, 2026.

iQOO. Supercomputing Chip Q2, 2024. URL https://www.iqoo.com/ae/en/support/questionByTitle?title=Supercomputing%20Chip%20Q2.

Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9000–9008, 2018.

Lingtong Kong, Boyuan Jiang, Donghao Luo, Wenqing Chu, Xiaoming Huang, Ying Tai, Chengjie Wang, and Jie Yang. IFRNet: Intermediate feature refine network for efficient frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1969–1978, 2022.

Zhen Li, Zuo-Liang Zhu, Ling-Hao Han, Qibin Hou, Chun-Le Guo, and Ming-Ming Cheng. AMT: All-pairs multi-field transforms for efficient frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9801–9810, 2023.

Liying Lu, Ruizheng Wu, Huaijia Lin, Jiangbo Lu, and Jiaya Jia. Video frame interpolation with transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3532–3542, 2022.

Junheum Park, Chul Lee, and Chang-Su Kim. Asymmetric bilateral motion estimation for video frame interpolation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14539–14548, 2021.

Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. FILM: Frame interpolation for large motion. In *Computer Vision – ECCV 2022*, pp. 250–266. Springer, 2022. doi: 10.1007/978-3-031-20071-7_15.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pp. 402–419. Springer, 2020.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. doi: 10.1109/TIP.2003.819861.

Sipeng Yang, Qingchuan Zhu, Junhao Zhuge, Qiang Qiu, Chen Li, Yuzhong Yan, Huihui Xu, Ling-Qi Yan, and Xiaogang Jin. Mob-FGSR: Frame generation and super resolution for mobile real-time rendering. In *ACM SIGGRAPH 2024 Conference Papers*, pp. 1–11. Association for Computing Machinery, 2024. doi: 10.1145/3641519.3657424.

Sipeng Yang, Jiayu Ji, Junhao Zhuge, Jinzhe Zhao, Qiang Qiu, Chen Li, Yuzhong Yan, Kerong Wang, Lingqi Yan, and Xiaogang Jin. Lightweight, edge-aware, and temporally consistent super-sampling for mobile real-time rendering. *ACM Transactions on Graphics*, 44(6):1–12, 2025. doi: 10.1145/3763348.

Guozhen Zhang, Yuhan Zhu, Haonan Wang, Youxin Chen, Gangshan Wu, and Limin Wang. Extracting motion and appearance via inter-frame attention for efficient video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5682–5692, 2023.

Guozhen Zhang, Haiguang Wang, Chunyu Wang, Yuan Zhou, Qinglin Lu, and Limin Wang. Arbitrary generative video interpolation. In *International Conference on Learning Representations*, volume 2026, pp. 31958–31977, 2026.

Zihao Zhang, Haoran Chen, Haoyu Zhao, Guansong Lu, Yanwei Fu, Hang Xu, and Zuxuan Wu. Eden: Enhanced diffusion for high-quality large-motion video frame interpolation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2105–2115. IEEE, 2025.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language polishing of the manuscript. All technical content, including the method design, experimental results, and analysis, was conducted and verified by the authors. All AI-assisted text was reviewed and revised by the authors, who take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

All experiments use the datasets, splits, and training configurations described in Section 4.1 and Appendix A. Baselines are reproduced from the authors' publicly released code, and metrics are computed with standard public implementations. Sections 3.2–3.3 describe the DMS operator and network architecture, and Section 4.1 specifies the training setup and objective.

A Implementation Details

Network architecture. The shared feature encoder applies three 3×3 convolutions with 16 channels and strides $2, 1, 1$, followed by a 4×4 transposed convolution with stride 2 to produce four-channel features at the network-input resolution. The four refinement blocks operate at scales $1/8, 1/4, 1/2, 1$, with channel widths $c = 72, 64, 48, 24$, respectively. Each block contains two 3×3 stride-two convolutions with $c/2$ and c output channels, followed by four residual units. Each residual unit applies a 3×3 convolution, a learnable per-channel scaling factor, a skip connection, and a LeakyReLU activation. All LeakyReLU activations use a negative slope of 0.2. A 4×4 stride-two transposed convolution followed by $2\times$ bilinear upsampling restores the stage resolution. The first three blocks output 14 channels: four flow residuals, one fusion-gate logit, one image-blending logit, and eight propagated feature channels. The last block outputs only the four flow residuals and two logits. The block input channel counts are $18, 32, 32, 32$: the coarsest block receives endpoint images, encoded features, and the geometric prior; finer blocks additionally use propagated flow, features, and the two logits from the preceding stage.

Motion propagation and fusion. At each finer stage, the propagated flow warps both endpoint images and their encoded features before refinement. To match Eq. (5), let $a^{(\ell)}$ be the predicted gate logit and σ the sigmoid function. We use $w^{(0)} = 1 - \sigma(a^{(0)})$ and $w^{(\ell)} = \sigma(a^{(\ell)})$ for $\ell = 1, 2, 3$. Thus, with $g^{(0)} = \mathbf{0}$, the coarsest block weights the prior by $\sigma(a^{(0)})$ before adding the residual. Finer blocks use the gate to blend propagated flow with the geometric prior. Gate and blending logits are propagated before sigmoid activation; the blending logit is converted to an image-fusion weight for synthesis.

Resampling and full-frame inference. Images, encoded features, and propagated states are resized bilinearly, whereas the geometric prior uses nearest-neighbor sampling. Motion values are scaled to the pixel units of each stage. At inference, the network processes the complete endpoint images at half resolution. The final flows and blending mask are then bilinearly upsampled to the original resolution, with flow values multiplied by two, to warp and blend the original images as in Eq. (6).

Optimization. We train with AdamW (zero weight decay) and distributed data parallelism, using $2{,}000$ linear warm-up steps followed by cosine learning-rate decay. Gradients are clipped to a norm of 1.0. The batch size is 36 per GPU, and the base peak learning rate is 1×10^{-4}, scaled linearly with the number of GPUs. Training runs for $2{,}000$ epochs on a server with Intel Xeon Gold 6430 CPUs and four NVIDIA L20 GPUs (48 GB each). Data augmentation includes random resizing and cropping, horizontal and vertical flips, 90° rotations, and random RGB channel permutations applied consistently across the frame triplet. Motion coordinates and vector components are transformed consistently with spatial augmentation.

Data preprocessing. Engine MVs are downsampled using 4×4 neighborhoods with stride 2. Each output location takes the MV and depth of the nearest surface within its neighborhood, and MV values are divided by two to match the half-resolution grid. The resulting motion is projected onto the target grid using DMS (Section 3.2), implemented with a custom CUDA kernel and depth-based collision handling.

B Additional Qualitative Results

Figure 6 presents additional qualitative comparisons across scenes and motion patterns. These examples complement the main-paper results, showing better preservation of scene structure and fewer artifacts in regions with large motion and complex visual effects.

The supplementary videos also compare Align-FG with DLSS 4 and other methods at inference, using the linear-based motion prior for Align-FG. They complement the still-image comparisons by showing temporal behavior during camera motion and visual effects.

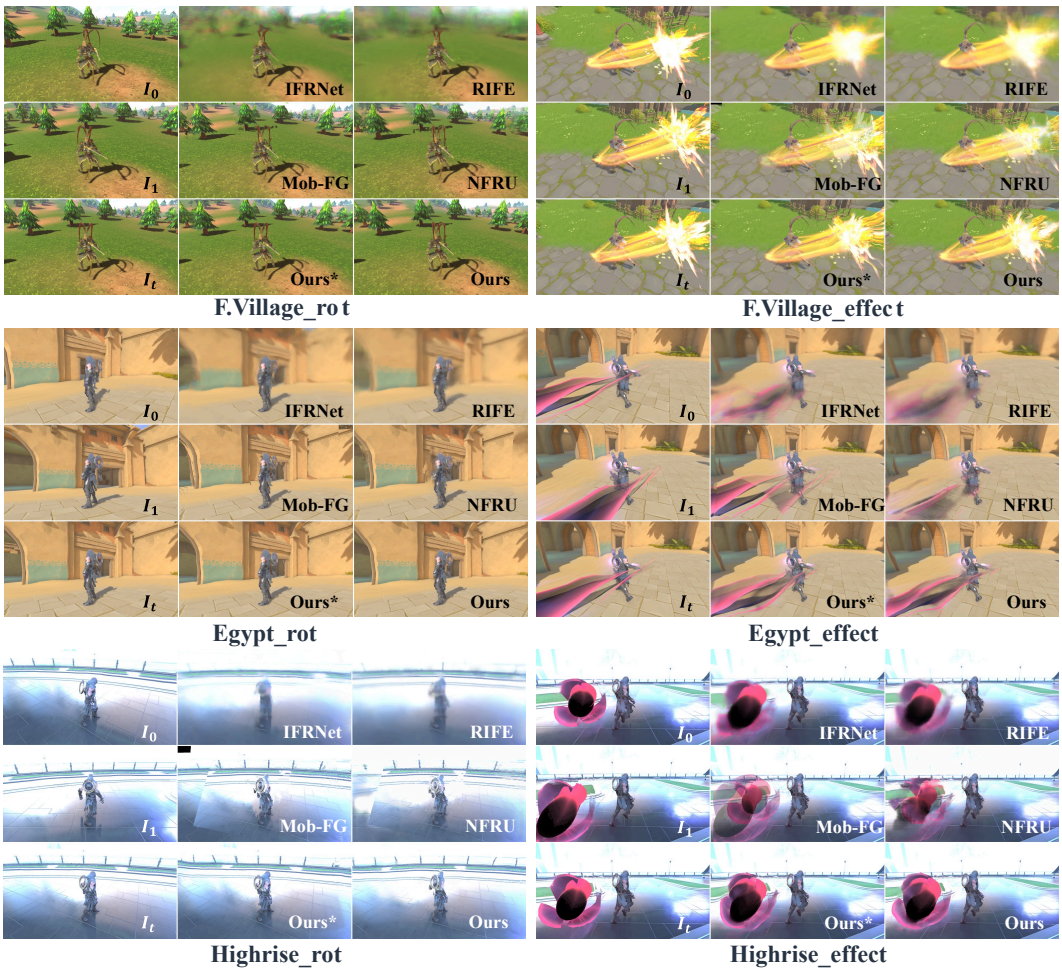


Figure 6: Additional qualitative comparisons across scenes and motion patterns. Align-FG better preserves scene structure and reproduces challenging motion and visual-effect regions closer to the ground truth.