

Our supplementary document includes two files:

- A video demo for dynamic visual comparisons between the proposed RDVFI-D and state-of-the-art interpolation methods, including: Wan (Wang et al., 2025), FCVG (Zhu et al., 2024), GI (Wang et al., 2024), MoMo (Lew et al., 2025), LDMVFI (Danier et al., 2024), and AMT (Li et al., 2023).
- This PDF file provides detailed comparisons with state-of-the-art methods, implementation details, and explanations of the proposed RDVFI.

A INFLUENCE OF KEYFRAME NUMBERS

This study examines the impact of keyframe number on interpolation quality using the FCVG dataset for $\times 24$ interpolation. Since RDVFI-D supports only 4N interpolation, RDVFI-U is chosen as the backbone model because of its flexibility. As indicated in Table 5, interpolation accuracy increases with the number of keyframes, as RDVFI can utilize functions with higher orders to resolve pixel trajectories and thus fit large and complex motions better. However, the marginal gain is observed beyond seven latent keyframes as the pixel trajectories have been advanced enough to model and resolve complex motions in VFI. The results demonstrate the effectiveness and efficiency of our RDVFI.

Table 5: Comparisons on keyframe numbers. The results show that the interpolation accuracy initially increases with the number of keyframes. More keyframes enable the proposed RDVFI to employ functions of higher order to solve large and complex motions, resulting in superior interpolation accuracy. The gain becomes marginal when the number of keyframes exceeds the default of seven frames, as the function is already sufficiently complex in solving motions for VFI.

Keyframes Number	LPIPS↓	FID↓	FVD↓
1	0.231	33.46	297.71
2	0.227	31.19	272.31
3	0.224	29.83	257.49
4	0.225	28.38	242.08
5	0.224	27.22	229.91
6	0.221	27.10	220.24
7 (default)	0.220	27.03	217.72
8	0.221	27.01	218.11
9	0.223	27.33	217.51

B NETWORK DETAILS

B.1 LATENT VIDEO DIFFUSION MODELS (LVDM)

We build our LVDMs for VFI upon SVD (Blattmann et al., 2023) and Wan2.1-Fun-1.3B-InP (Wang et al., 2025) models. We do not apply LoRA (Hu et al., 2022) fine-tuning technique for both models because supervised fine-tuning leads to significant performance gain. To avoid overfitting and backbone model degradation through fine-tuning, we collect and train our diffusion on a large-scale dataset that consists of more than 1 million valid video clips with random data augmentation, like horizontal flipping, random resizing, and cropping. Because the Wan2.1-Fun-1.3B-InP model supports start-end-frame generation, we start with the default network to implement LVDM in our RDVFI-D. We simply adapt image-to-video generation SVD (Blattmann et al., 2023) by concatenating the input frames along the temporal dimension and setting their diffusion timesteps to the least value that yields a noise-free input. We partially initialize the pretrained stable-video-diffusion-img2vid-xt-1-1 weights to implement LVDM in RDVFI-U.

B.2 CONTINUOUS PIXEL TRAJECTORIES ESTIMATOR

Our continuous intermediate motion field estimator consists of two networks: $\phi_1^f(\cdot)$ and $\phi_2^f(\cdot)$. The detailed network working flow and structures are shown in Algorithm 2 and Table 6, respectively.

Algorithm 2: Continuous Pixel Trajectories Estimator

Require: Denoised latent features $\{z_{\tau_i}\}_{i=1}^N$; encoded latent frames z_0, z_1 ; Two neural networks ϕ_1^f, ϕ_2^f

Ensure: Flows for each interpolation $\{f_{0 \rightarrow \tau_i}, f_{1 \rightarrow \tau_i}\}_{i=1}^N$

- 1: Initialize $z_{\tau_0} = z_0, z_{\tau_{N+1}} = z_1$
- 2: Calculate $f_{\tau_i \rightarrow \tau_{i+1}} = \phi_1^f(z_i, z_{i+1})$;
- 3: **for** $i=0, 1, \dots, N$ **do**
- 4: Warp z_0 with previous flow estimation results $z_{0 \rightarrow \tau_{i-1}} = \text{Warp}(f_{0 \rightarrow \tau_{i-1}}, z_0)$;
- 5: Flow fusion for time $\tau = \tau_i$: $f_{0 \rightarrow \tau_i} = \phi_2^f(z_0, z_{\tau_i}, z_{0 \rightarrow \tau_{i-1}}, f_{0 \rightarrow \tau_{i-1}}, f_{\tau_{i-1} \rightarrow \tau_i}) + f_{0 \rightarrow \tau_{i-1}}$
- 6: **end for**
- 7: **for** $i=N, N-1, N-2, \dots, 1$ **do**
- 8: Warp z_1 with previous flow estimation results $z_{1 \rightarrow \tau_{i+1}} = \text{Warp}(f_{1 \rightarrow \tau_{i+1}}, z_1)$;
- 9: Flow fusion for time $\tau = \tau_i$: $f_{1 \rightarrow \tau_i} = \phi_2^f(z_1, z_{\tau_i}, z_{1 \rightarrow \tau_{i+1}}, f_{1 \rightarrow \tau_{i+1}}, f_{\tau_{i+1} \rightarrow \tau_i}) + f_{1 \rightarrow \tau_{i+1}}$
- 10: **end for**

We use the same $\phi_1^f(\cdot), \phi_2^f(\cdot)$ for estimation (same network, same weights) for all intermediate frames and both flow fusion directions.

B.3 FRAME SYNTHESIS

We synthesize intermediate frames with estimated optical flows by borrowing IFBlocks from RIFE (Huang et al., 2022), shown in Table 7. The frame synthesis network $\phi^s(\cdot)$ two IFBlocks at $\frac{1}{4}$ and $\frac{1}{2}$ scales, respectively. We warp frame using refined optical flows, resulting in $I_{0 \rightarrow \tau_j}, I_{1 \rightarrow \tau_j}$, respectively, and fuse with the mask m by:

$$I_{\tau_j} = \text{sigmoid}(m) \cdot I_{0 \rightarrow \tau_j} + (1 - \text{sigmoid}(m)) \cdot I_{1 \rightarrow \tau_j}. \quad (6)$$

where I_{τ_j} is the interpolated frame.

As shown in Table 8, we then refine the fused intermediate frame $\hat{I}_{\tau_j}$ with denoised latent features for variations that cannot be modeled by flows. We first detach the gradient of denoised keyframe latent features $\{z_{k_i}, z_{k_{i+1}}\}$, where $z_{k_i} < \tau_j < z_{k_{i+1}}$ to force the frame synthesis network interpolate with correct motion rather than directly decoding latent features during training. Then, we warp these latent features with sampled optical flow $\{f_{k_i \rightarrow \tau_j}, f_{k_{i+1} \rightarrow \tau_j}\}$. We upscale the warped latent to the same resolution as the input frames, following a resblock at 64 channels. Finally, we add them to the fused intermediate frame to generate the final interpolation results $\hat{I}_{vfi}$ at time $\tau = \tau_j$.

C ADDITIONAL IMPLEMENTATION DETAILS

C.1 DATASETS AND EVALUATION METRICS

Following (Zhu et al., 2024), we form our training dataset by filtering video clips from the DAVIS dataset (Pont-Tuset et al., 2017), the RealEstate10K dataset (Zhou et al., 2018), supplemented by high-frame-rate videos from Pixels¹. We filtered out video clips that contain abnormal motions like static or scene changes according to optical flows across consecutive frames. Each training sample contains 25 frames and has a spatial resolution of 1280×720 , similar to (Zhu et al., 2024). We evaluate our methods on two benchmark datasets, the DAVIS-7 dataset by Jain et al. (2024) for $\times 8$ interpolation, the evaluation dataset by Zhu et al. (2024) for $\times 24$ interpolation, as they contain diverse motion patterns and objects. Additionally, to ensure data diversity, we created an evaluation set comprising 52 human-selected, high-quality videos from Pixels, producing 100 video clips for $\times 24$ interpolation at a resolution of 1024×576 . All of the selected datasets are well-designed and contain large and complex motions. We report numeric comparison results on both reconstruction metrics, SSIM, and perceptual metrics, including LPIPS (Zhang et al., 2018), FID (Heusel et al., 2017), and FVD (Ge et al., 2024). We introduce FID and FVD for evaluation, as they assess distribution distances between prediction results and ground truth, which effectively evaluate VFI quality in large, complex motions with various motion patterns.

¹<https://www.pexels.com/>

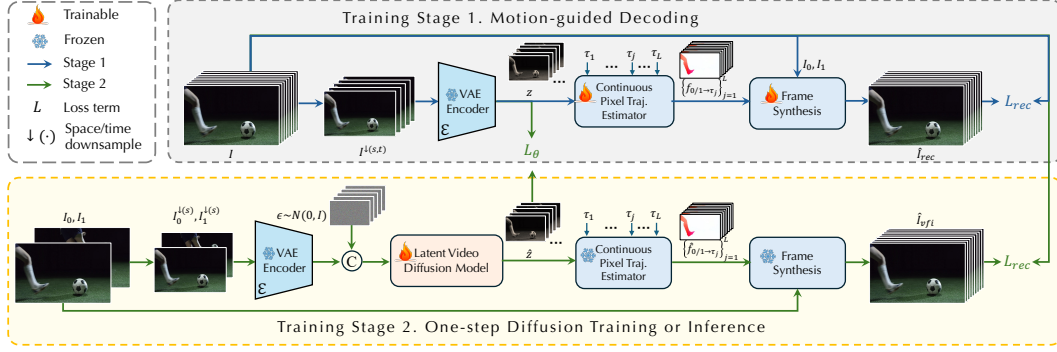


Figure 8: Overall framework of the proposed RDVFI. Our RDVFI interpolates intermediate frames $\hat{I}_{vfi}$ using one-step diffusion sampling with input two ending frames I_0, I_1 . For efficient and effective training, we propose a two-stage training strategy. **Stage 1** trains a motion-guided decoding that decodes the VAE latent into continuous pixel trajectories, which are used to synthesize high-resolution frames. **Stage 2** trains a one-step diffusion model to generate latent features of intermediate frames from two input frames. Its output will be used as input to the motion estimator trained in stage 1.

C.2 MORE TRAINING DETAILS

We show our two-stage training scheme in Fig. 8 and provide more details as follows. Please find Section 3.5 for more explanations and details.

Motion-guided Decoding As introduced in Section B.3, our RDVFI renders final interpolated frames based on both warped frames and latent keyframes to approximate variations that cannot be approximated using the proposed continuous pixel trajectories, such as dynamic textures and illumination changes. To avoid RDVFI, which interpolates mainly by warped latent keyframes rather than the input frames, we utilize fixed zero tensors to replace warped for the first 200K iterations of training. Then, we feed the network with the actual warped latent keyframes and fine-tune the network at a learning rate of $5e-6$ for another 200K iterations.

LVDM Training Our LVDM training involves a warm-up process, gradually increasing the lower bound for diffusion timestep sampling. This process takes 80K iterations. Then, the LVDM in RDVFI learns to predict latent keyframes from pure noise through training for the left 720K iterations.

D MORE VISUALIZATION

We present more visualization comparisons in Fig. 9, Fig. 10, and Fig. 11.

Non-generative Methods (Huang et al., 2022; Reda et al., 2022; Li et al., 2023; Guo et al., 2024) cannot handle large and complex motions as they rely on regression-based training. Thus, they tend to introduce severe ghosting artifacts that sometimes result in disappearing moving objects (the girl in Fig. 10).

Image Diffusion-based Methods (Danier et al., 2024; Lew et al., 2025) attempt to mitigate motion ambiguities by involving an image diffusion model to guide motion estimation. Despite their effectiveness in advancing VFI with small and simple motions, they cannot deal with large and complex motions either. The LDMVFI (Danier et al., 2024) estimates optical flows based on invalid latent frame estimation, thus producing optical flows that wrap pixels incorrectly, resulting in distorted objects. Unlike LDMVFI (Danier et al., 2024), MoMo (Lew et al., 2025) directly generates optical flows using an image diffusion model, which uses one of the input frames as the result when motions are extremely challenging.

Video Diffusion-based Methods (Zhu et al., 2024; Wang et al., 2024; 2025) directly generate intermediate frames with Latent Video Diffusion Models (LVDMs), producing frames with rich details. However, the interpolated contents and motions are inconsistent across frames as shown in Fig. 11,

Table 6: Network implementation details of the continuous pixel trajectories estimator networks. Convolution specifications are given in order kernel height $\times$ kernel width $\times$ output channel; negative slope of LeakyReLU is in the parentheses. We take forward flow fusion as an example and can estimate backward flow fusion results using the same network in reverse order.

Number	Operation	Input
ϕ_1		
1	Concatenate	z_{k_i}, z_{k_i+1}
2	Conv $3 \times 3 \times 256$	#1
3	Conv $3 \times 3 \times 256$	#2
4	LeakyReLU(0.2)	#3
5	Conv $3 \times 3 \times 256$	#4
6	LeakyReLU(0.2)	#5
7	Conv $3 \times 3 \times 256$	#6
8	LeakyReLU(0.2)	#7
9	Add	#2, #8
10	Conv $3 \times 3 \times 2$	#9
ϕ_2		
11	Warp	#10, z_0
12	Concatenate	#11, $z_0, z_{k_i+1}, f_{0 \rightarrow k_i}, \#10$
13	Conv $3 \times 3 \times 256$	#12
14	LeakyReLU(0.2)	#13
15	Conv $3 \times 3 \times 256$	#14
16	LeakyReLU(0.2)	#15
17	Conv $3 \times 3 \times 256$	#16
18	LeakyReLU(0.2)	#17
19	Add	#13, #18
20	Conv $3 \times 3 \times 2$	19
21	Add	#20, $f_{0 \rightarrow k_i+1}$

Table 7: Network implementation details of the IFBlock networks. Convolution specifications are given in order kernel height $\times$ kernel width $\times$ output channel; negative slope of LeakyReLU is in the parentheses. Convolution channel C varies with resize scales, which are 256 and 128 for 0.25 and 0.5 scales, respectively. We take forward flow fusion as an example and can estimate backward flow fusion results using the same network in reverse order.

#	Operation	Input
1	Warp	$I_0, f_{0 \rightarrow \tau_j}$
2	Warp	$I_1, f_{1 \rightarrow \tau_j}$
3	Concatenate	$m, \#1, \#2, I_0, I_1, f_{0 \rightarrow \tau_j}, f_{1 \rightarrow \tau_j}$
4	Resize	#3
5	Conv $3 \times 3 \times C$	#4
6	LeakyReLU(0.2)	#5
7	Conv $3 \times 3 \times C$	#6
8	LeakyReLU(0.2)	#7
9	Conv $3 \times 3 \times C$	#8
10	LeakyReLU(0.2)	#9
11	Add	#10, #5
12	Conv $3 \times 3 \times 5$	#11
13	Add	First two channels of #12, $f_{0 \rightarrow \tau_i}$
14	Add	Following two channels of #12, $f_{1 \rightarrow \tau_i}$
15	Add	Last channel of #12, m

resulting in object fractions (see foot regions), ghosting effects (skateboard), and unnatural motions (koala).

Our RDVFI solves pixel movements with high-order functions, which can accurately capture real-world dynamics. As a result, the proposed RDVFI consistently outperforms existing methods across different scenarios.

Table 8: Full network implementation details of the frame synthesis network.

#	Operation	Input
1	IFBlock at 1/4 scale	$I_0, I_1, f_{0 \rightarrow \tau_j}, f_{1 \rightarrow \tau_j}$
2	IFBlock at 1/2 scale	#1, I_0, I_1
3	Gradient Detach	z_{k_i}
4	Gradient Detach	$z_{k_{i+1}}$
5	Warp	#3, $f_{k_i \rightarrow \tau_j}$
6	Warp	#4, $f_{k_{i+1} \rightarrow \tau_j}$
7	Concatenate and Upscale	#3, #4
8	Conv $3 \times 3 \times 64$	#7
9	Conv $3 \times 3 \times 64$	#8
10	LeakyReLU(0.2)	#9
11	Conv $3 \times 3 \times 64$	#10
12	LeakyReLU(0.2)	#11
13	Conv $3 \times 3 \times 64$	#12
14	LeakyReLU(0.2)	#13
15	Add	#8, #14
16	Conv $3 \times 3 \times 3$	#15

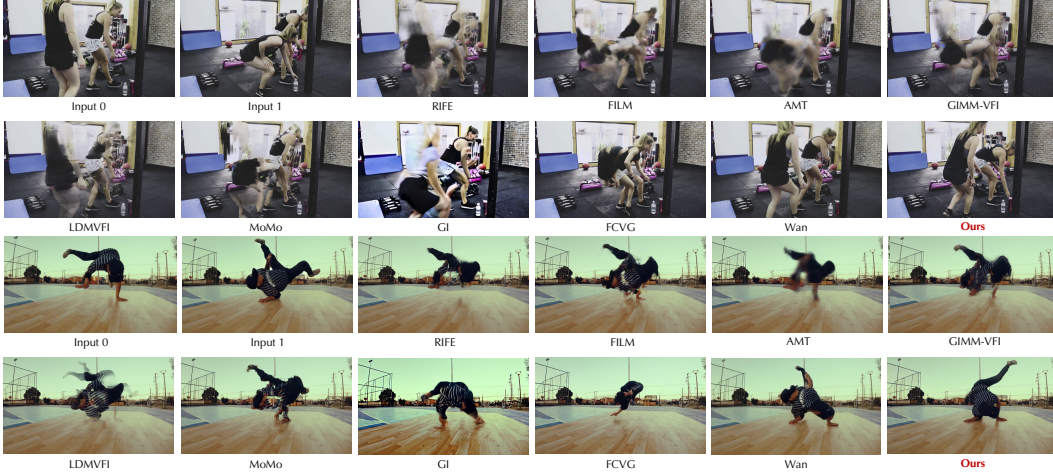


Figure 9: Visual comparisons on the FCVG (Zhu et al., 2024) dataset (top row) and our self-collected dataset (bottom row).

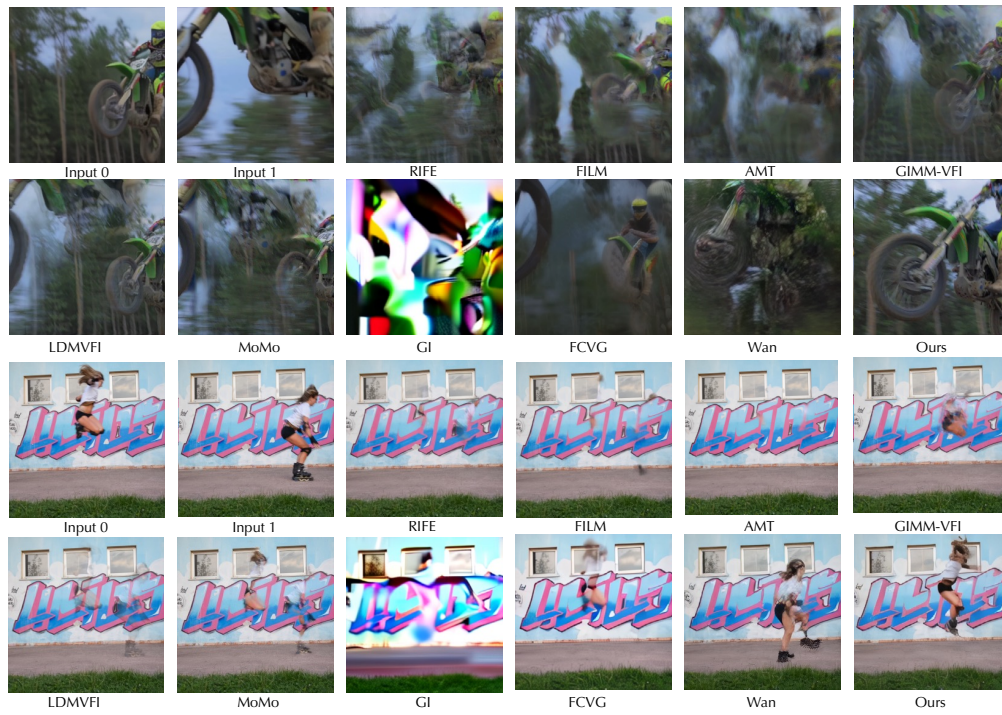


Figure 10: Visual comparisons on DAVIS-7 (Jain et al., 2024) dataset.



Figure 11: Sequence visual comparisons on our self-collected dataset.