Splatting-based Synthesis for Video Frame Interpolation

Simon Niklaus
Adobe Research

Ping Hu
Boston University

Jiawen Chen
Adobe Inc

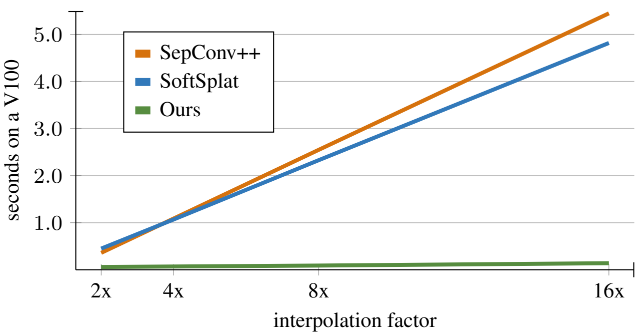


Figure 1. Runtime of two common video frame interpolation approaches versus ours when interpolating multiple frames between two inputs from XTEST-2K [56]. Our proposed approach interpolates the first frame in 61 ms and each additional frame only takes a few milliseconds thanks to our splatting-based synthesis.

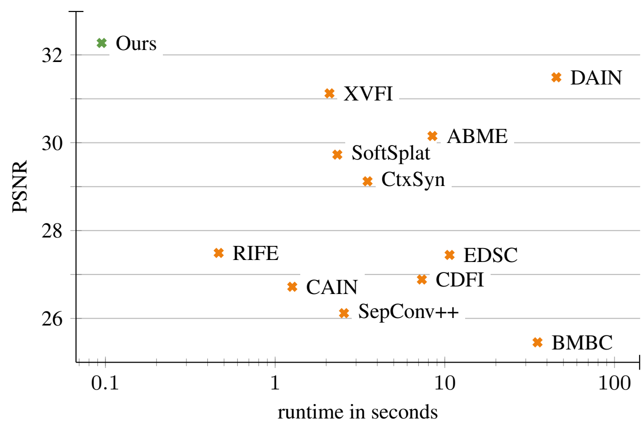


Figure 2. Evaluating the $8\times$ interpolation ability of our proposed approach in comparison to various others on XTEST-2K [56].

Abstract

Frame interpolation is an essential video processing technique that adjusts the temporal resolution of an image sequence. While deep learning has brought great improvements to the area of video frame interpolation, techniques that make use of neural networks can typically not easily be deployed in practical applications like a video editor since they are either computationally too demanding or fail at high resolutions. In contrast, we propose a deep learning approach that solely relies on splatting to synthesize interpolated frames. This splatting-based synthesis for video frame interpolation is not only much faster than similar approaches, especially for multi-frame interpolation, but can also yield new state-of-the-art results at high resolutions.

1. Introduction

Video frame interpolation is becoming more and more ubiquitous. While early techniques for frame interpolation were restricted to using block motion estimation and compensation due to performance constraints [8, 20], modern graphics accelerators allow for dense motion estimation and compensation while heavily making use of neural networks [36, 44, 45, 47]. These developments enable interesting new applications of video frame interpolation for animation inbetweening [31], video compression [62], video editing [39], motion blur synthesis [3], and many others.

However, current interpolation techniques that make use of neural networks are inherently difficult to accelerate. For example, the first interpolation approaches that use deep learning require fully executing the entire network for each output [36, 44, 45]. As such, using SepConv++ [46] (Figure 1, orange) to interpolate a video by a factor of $8\times$ instead of $2\times$ requires eight times more compute. Newer approaches are little different though, SoftSplat [43] (Figure 1, blue) for instance estimates the optical flow between the input frames and then extracts and warps feature pyramids to the desired instant before employing a synthesis network to yield the final result. While the optical flow only needs to be estimated once in this case, the synthesis network has to be executed for each new frame which again requires roughly eight times more compute when interpolating by $8\times$ instead of $2\times$.

To address such limitations, we propose a splatting-based synthesis approach. Specifically, we propose to solely rely on splatting to synthesize the output image without any subsequent refinement. As such, interpolating frames after estimating the optical flow requires only a few milliseconds and interpolating a video by a factor of $8\times$ instead of $2\times$ requires hardly any more compute thanks to our image forma-

(a) overlaid inputs (b) SoftSplat [43] (c) XVFI [56] (d) Ours

Figure 3. Qualitative comparison of our proposed approach with two representative methods on a sample from the XTEST-2K [56] test dataset. While these sophisticated interpolation methods are unable to handle this challenging scenario with the utility pole subject to large motion, our comparatively simple approach is able to generate a plausible result. Please consider our supplementary for more results.

tion model (Figure 1, green). Further, our synthesis approach allows for the motion to be estimated at a lower resolution and to then upsample the estimated flow before using it to warp the input frames. This not only improves the computational efficiency, but can counterintuitively also lead to an improved interpolation quality (Figure 2 and Figure 3).

The key to making our splatting-based synthesis approach work well is that it is carefully designed and that it is fully differentiable. Our careful design greatly improves the interpolation quality when compared to a common optical flow warping baseline ($+1.35$ dB on Vimeo-90k [65]), and being fully differentiable enables the underlying optical flow estimator to be fine-tuned which further improves the interpolation results ($+1.43$ dB on Vimeo-90k [65]). Summarizing our claims in short, we (1) introduce an image synthesis approach purely based on splatting that is especially well-suited for multi-frame interpolation, (2) show that iterative optical flow upsampling not only further improves the efficiency of our approach but can also lead to an improved quality, and (3) identify a numerical instability in softmax splatting and propose an effective solution to address it.

2. Related Work

Warping-based frame interpolation has a long history. Some examples based on block-level motion estimates include overlapping block motion compensation [8, 20], adaptively handling overlapping blocks [7], detecting and handling occlusions [24], considering multiple motion estimates [27], and estimating a dense motion field at the interpolation instant [12]. These are in contrast to motion compensation based on dense estimates which includes layered warping [53, 70], occlusion reasoning for temporal interpolation [22], transition points [38], and using warping as a metric to evaluate optical flow estimates [1].

Our proposed splatting-based synthesis is closely related to traditional warping techniques that leverage optical flow estimates while reasoning about occlusions [1, 22]. However, for a splatting-based synthesis approach to be used in a deep learning setting, the involved operations needs to be differentiable and easy to parallelize. This prohibits common techniques such as ordering and selecting a candidate flow in cases where multiple source pixels map to the same target [22], or iteratively filling holes [1]. In contrast, our proposed splatting-based synthesis technique only relies on differentiable operations that are easy to parallelize such as softmax splatting [43] and backward warping [26].

A common category of frame interpolation approaches interpolate a frame at an arbitrary time t between two input frames. We have summarized recent techniques from this category in the supplementary material since these are most closely related to our proposed approach. All of these methods have in common that they require running a neural network to infer the interpolation result at the desired instant. That is, they either use a neural network to refine warped representations of the input images, or use a neural network to infer the motion from the desired interpolation instant to the input images before accounting for it. Running such neural networks is computationally challenging though, especially at high resolutions. This is in contrast to our splatting-based synthesis where, given optical flow estimates between the input frames, synthesizing the interpolation result at any time instant requires only a few primitive operations.

Another category of video frame interpolation approaches take two images as input and interpolate a frame at a fixed time, typically $t = 0.5$, between the two inputs. This includes kernel-based synthesis techniques [44, 45, 46], approaches that estimate the motion from the frame that is ought to be interpolated either implicitly [5, 30, 55] or explicitly [19, 35, 36, 50, 51, 67, 68], methods that directly synthesize the result [10, 28], and techniques that estimate the phase decomposition of the intermediate frame [40]. We focus on arbitrary-time video frame interpolation.

The area of frame interpolation is much more diverse than

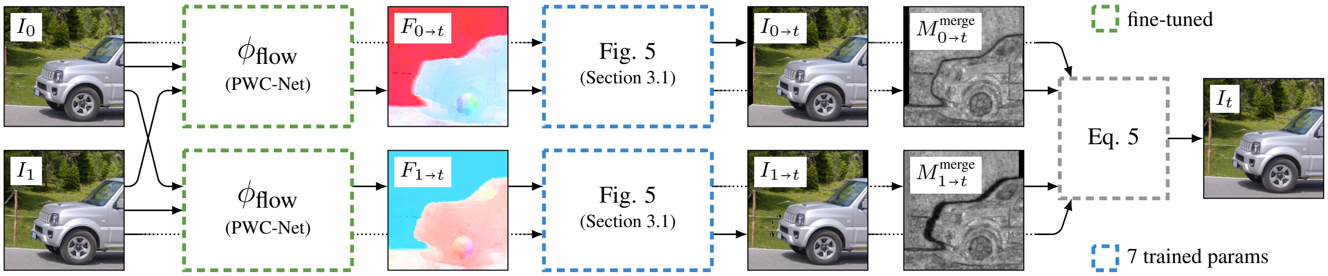


Figure 4. Overview of our proposed splatting-based synthesis for video frame interpolation. Given two frames I_0 and I_1, we estimate the inter-frame motion $F_{0\to1}$ and $F_{1\to0}$ through an off-the-shelf optical flow network. Using the flow scaled by the desired instant t, we then splat the input frames to time t as $I_{0\to t}$ and $I_{1\to t}$ as outlined in Figure 5 before merging them according to Equation 1 to obtain I_t.

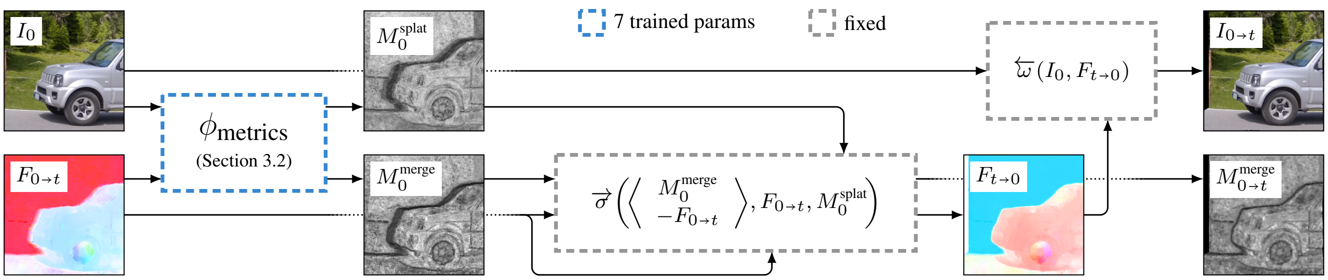


Figure 5. Given an image I_0 as well as an optical flow $F_{0\to t}$, we not only splat the image to time t as $I_{0\to t}$ but also a generate a corresponding weight map $M^{\text{merge}}_{0\to t}$ that can be used to merge multiple synthesis results. Specifically, we use softmax splatting $\overrightarrow{\sigma}$ [43] to splat inverse flows before employing backward warping $\overleftarrow{\omega}$ [26] to reconstruct $I_{0\to t}$ from I_0 and we directly splat a base metric M_0^{merge} to obtain $M^{\text{merge}}_{0\to t}$.

these categories though. There is research on using multiple input frames [6, 34, 54, 64], interpolating footage from event cameras [33, 59, 61, 66], efficient model design [10, 11, 13], test-time adaptation [9, 52], hybrid imaging systems [48], handling quantization artifacts [60], as well as joint deblurring [54] and super-resolution [29, 63]. Our splatting-based synthesis is orthogonal to such research directions.

3. Splatting-based Synthesis

Our proposed splatting-based synthesis approach for video frame interpolation is summarized in Figure 4 and we will subsequently discuss its individual aspects. In doing so, we consider (1) how to resolve ambiguities where multiple pixels from the input image map to the same location in the target, (2) how to do the warping without introducing any unnecessary artifacts, and for video frame interpolation in particular (3) how to merge I_0 and I_1 after warping them to synthesize the desired interpolation result I_t at time t.

3.1. Splatting and Merging

The core of our splatting-based synthesis is to warp I_0 and I_1 to the desired interpolation instant t using $F_{0\to t}$ and $F_{1\to t}$ respectively. However, one cannot simply splat an input image as is since multiple pixels in the source image may map to the same target location as shown in Figure 6. To address this ambiguity, we follow [43] and use an auxiliary weight M^{splat} that serves as a soft inverse z-buffer (called Z in [43]). We discuss how to obtain M^{splat} in Section 3.2.

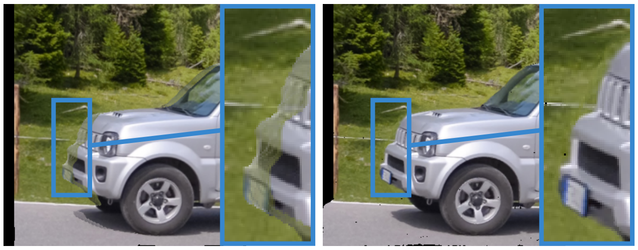
One may be tempted to directly splat I_0 using the optical flow $F_{0\to t}$ subject to the splatting metric M_0^{splat} in order to obtain $I_{0\to t}$ (I_0 warped to time t). However and as shown in Figure 7, this naive application of softmax splatting will lead to subtle artifacts and introduce unnecessary blurriness. Instead, we follow existing warping-based interpolation approaches and splat $F_{0\to t}$ to t to obtain the inverse flow $F_{t\to0}$ which is then used to backward warp I_0 to t [1, 22].

Splatting naturally leads to holes in the warped result due to not only occlusions but also divergent flow fields. As shown in Figure 8, splatting with a divergent flow results in small holes even in contiguous areas. To fill these holes, we replace the default bilinear splatting kernel, which only has a footprint of 2×2, with a 4×4 Gaussian kernel. Note that such a wider kernel would lead to blurrier results when splatting colors, but it does not affect the clarity in our approach where we splat inverse flows and then backward warp the image.

After these careful considerations we are able to faithfully warp I_0 to $I_{0\to t}$ and I_1 to $I_{1\to t}$, but we cannot simply average these individual results to obtain the desired I_t since some pixels are more reliable than others as shown in Figure 9. As such, we introduce an auxiliary map M^{merge} that weights the individual results before merging them to obtain I_t as:

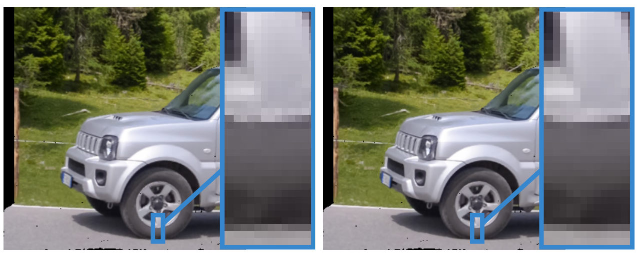
$$I_t = \frac{(1-t)\cdot M^{\text{merge}}_{0\to t}\cdot I_{0\to t} + t\cdot M^{\text{merge}}_{1\to t}\cdot I_{1\to t}}{(1-t)\cdot M^{\text{merge}}_{0\to t} + t\cdot M^{\text{merge}}_{1\to t}} \quad (1)$$

where $I_{0\to t}$ is I_0 warped to time t, $M^{\text{splat}}_{0\to t}$ is M_0^{splat} warped to time t, and analogous for $I_{1\to t}$ and $M^{\text{splat}}_{1\to t}$ in the opposite



(a) naively splat I_0 to get $I_{0\to t}$ (b) splatting weighted by M_0^{splat}

Figure 6. We use a splatting metric M^{splat} that weights the individual pixels to resolve ambiguities where multiple pixels map to the same destination, thus properly handling occlusions.



(a) splat colors directly (b) splat flows then backwarp colors

Figure 7. Directly splatting the colors of an image can lead to subtle artifacts, which is why we splat flows instead and then synthesize the output using backwards warping of the splatted flows.

direction. We will subsequently describe how to obtain the involved splatting M^{splat} and merging M^{merge} metrics.

3.2. Metrics for Splatting and Merging

Previous frame interpolation work used photometric consistency to resolve the splatting ambiguity where multiple source pixels map to the same target location [1]. This measure can be defined using backward warping $\overleftarrow{\omega}(\cdot)$ as:

$$\psi_{\text{photo}} = \left\| I_0 - \overleftarrow{\omega}\left(I_1, F_{0\to 1}\right) \right\| \tag{2}$$

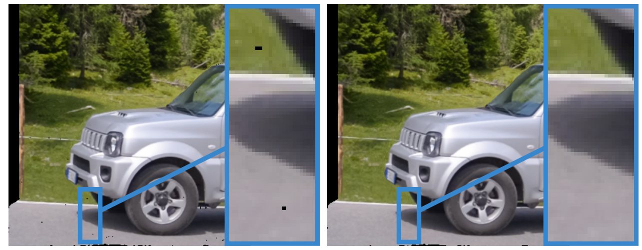
However, photometric consistency is easily affected by brightness changes, as is frequently the case with moving shadows. As such, we not only consider photometric consistency but also optical flow consistency defined as:

$$\psi_{\text{flow}} = \left\| F_{0\to 1} + \overleftarrow{\omega}\left(F_{1\to 0}, F_{0\to 1}\right) \right\| \tag{3}$$

Flow consistency is given if the flow of a pixel mapped to the target maps back to the pixel in the source, which is invariant to brightness changes. Another measure we consider is flow variance, which indicates local changes in flow as:

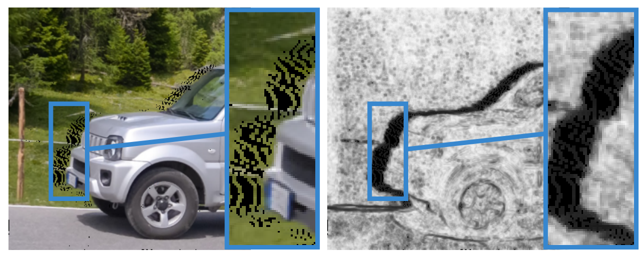
$$\psi_{\text{varia}} = \left\| \sqrt{G({F_{0\to 1}}^2) - G(F_{0\to 1})^2} \right\| \tag{4}$$

where $G(\cdot)$ denotes a 3×3 Gaussian filter. Flow variance is high in areas with discontinuous flow, as is the case at motion boundaries. As shown in Figure 9, optical flow estimates tend to be inaccurate at boundaries which makes this measure particularly useful for the M^{merge} metric.



(a) splat flows with bilinear kernel (b) splat flows with Gaussian kernel

Figure 8. Splatting is subject to holes not only due to occlusions but also due to divergent flow fields, which we address by replacing the bilinear splatting kernel with a wider Gaussian kernel.



(a) warped image $I_{1\to t}$ (b) corresponding $M_{1\to t}^{\text{merge}}$

Figure 9. We use a merging metric M^{merge} that weights the individual pixels in the warped images $I_{0\to t}$ and $I_{1\to t}$, which suppresses the influence of unreliable pixels when generating I_t.

	Middlebury Baker *et al.* [1]		Vimeo-90k Xue *et al.* [65]	
	PSNR ↑	absolute change	PSNR ↑	absolute change
Ours	36.63	–	35.00	–
w/o flow splatting	36.27	- 0.36 dB	34.86	- 0.14 dB
w/o gaussian splatting	36.39	- 0.24 dB	34.89	- 0.11 dB
w/o stable splatting	36.48	- 0.15 dB	34.97	- 0.03 dB
w/o using ψ_{photo}	36.22	- 0.41 dB	34.99	- 0.01 dB
w/o using ψ_{flow}	36.44	- 0.19 dB	34.99	- 0.01 dB
w/o using ψ_{varia}	36.40	- 0.23 dB	34.89	- 0.11 dB

Table 1. Ablative experiments to analyze the design choices of our proposed splatting-based synthesis for video frame interpolation.

We conclude by combining these measures and define the splatting M^{splat} metric as (and analogous for M^{merge}):

$$M^{\text{splat}} = \frac{1}{1+\alpha_{\text{p}}^{\text{s}}\cdot\psi_{\text{photo}}} + \frac{1}{1+\alpha_{\text{f}}^{\text{s}}\cdot\psi_{\text{flow}}} + \frac{1}{1+\alpha_{\text{v}}^{\text{s}}\cdot\psi_{\text{varia}}} \tag{5}$$

where $\left\langle \alpha_{\text{p}}^{\text{s}}, \alpha_{\text{f}}^{\text{s}}, \alpha_{\text{v}}^{\text{s}} \right\rangle$ are tuneable parameters. The merge metric M^{merge} is defined analogous with $\left\langle \alpha_{\text{p}}^{\text{m}}, \alpha_{\text{f}}^{\text{m}}, \alpha_{\text{v}}^{\text{m}} \right\rangle$. We also scale M^{splat} by an α as in [43], and initially set these seven parameters to 1 while learning their values through end-to-end training. We also tried using a neural network to merge the individual measures, but have found Equation 5 to be faster and work better. Lastly, we also considered more com-

	Middlebury Baker *et al.* [1]		Vimeo-90k Xue *et al.* [65]		Xiph-1K (4K scaled to 1K)		Xiph-2K (4K scaled to 2K)		Xiph-4K (from xiph.org)	
	PSNR ↑	relative change	PSNR ↑	relative change	PSNR ↑	relative change	PSNR ↑	relative change	PSNR ↑	relative change
fixed PWC-Net w/ [1] warping	33.80	–	32.22	–	33.61	–	33.59	–	32.61	–
fixed PWC-Net w/ our warping	34.73	+ 0.93 dB	33.57	+ 1.35 dB	35.03	+ 1.42 dB	34.90	+ 1.31 dB	33.66	+ 1.05 dB
tuned PWC-Net w/ our warping	36.63	+ 1.90 dB	35.00	+ 1.43 dB	36.75	+ 1.72 dB	35.95	+ 1.05 dB	33.93	+ 0.27 dB

Table 2. Comparing our splatting-based synthesis to a common warping-based interpolation technique [1]. Not only does our approach greatly outperform this baseline, it also allows us to fine-tune the utilized PWC-Net [57] which further improves the interpolation results.

plex measures such as depth [2] but have found these not to be beneficial due to their computational complexity.

3.3. Ablative Experiments

We analyze the choices we made when designing our splatting-based synthesis for frame interpolation through ablative experiments. As shown in Table 1, each individual component contributes to the interpolation quality.

3.4. Baseline Comparison

We compare our proposed splatting-based synthesis for frame interpolation to a common warping-based interpolation technique [1] in Table 2, which shows that our approach greatly outperforms this common baseline. However, since our image formation model is end-to-end differentiable, we can further improve the quality of our interpolated results by fine tuning the underlying optical flow estimator. Essentially, we show how to perform the technique of [1] better and in a differentiable manner to enable end-to-end supervision.

3.5. Real-time Interpolation

Our splatting-based synthesis allows synthesizing a frame within a few milliseconds once the inter-frame motion has been estimated. We demonstrate this ability through an interactive visualization tool that is provided in the supplementary material (see Figure 10). This demo takes two images as well as pre-computed optical flow estimates as input and essentially implements Figure 5 as well as Equation 1 to synthesize the interpolated frame at the requested instant. This visualization is implemented in Javascript and it neither uses multi-threading nor any graphics acceleration. Despite this naive implementation, the demo is still able to interpolate frames in real time thanks to our image formation model.

4. Iterative Flow Upsampling

It is impractical to compute optical flow on a 4K video. For high-resolution inputs, we thus propose to estimate the motion at a lower resolution and then use a neural network to iteratively upsample the optical flow to the full resolution of the input (see Figure 11). In practice, one may want to estimate the optical flow on either a 2K or a 1K resolution when given a 4K video depending on the desired performance

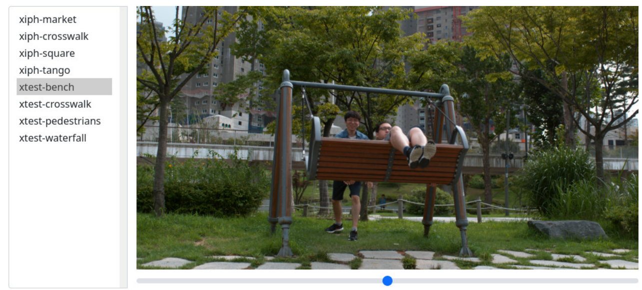


Figure 10. An interactive demo which performs our splatting-based synthesis on the fly, please see the supplementary "visualization.html". This is a video that is best viewed in Adobe Reader.

characteristics. To support this use case, we subsequently propose an iterative optical flow upsampling approach.

4.1. Iterative Upsampling

We utilize a small neural network to perform iterative flow upsampling in an coarse-to-fine manner while using the high-resolution input frames as a guide. Specifically, given a flow estimate at a resolution of $\boldsymbol{x}$ as well as the two input images at a resolution of $2 \cdot \boldsymbol{x}$, the upsampling network estimates the flow at a resolution of $2 \cdot \boldsymbol{x}$ through a sequence of four convolutions with PReLU [21] activations in between. To upsample a given optical flow estimate by a factor of $4\times$, we execute the upsampling network twice.

We have found it beneficial to not only guide the upsampling by providing the input images, but also the three measures from Section 3.2 as they encode useful properties of the optical flow. We have otherwise kept our upsampling network deliberately simple without using spatially-varying upsampling kernels [58], normalized convolution upsampling [15], or self-guided upsampling [37]. After all, one of the reasons for estimating the optical flow at a lower resolution is improved efficiency and employing a more complex upsampling network would counteract this objective.

Another reason for estimating the optical flow a lower resolution is to mimic the inter-frame motion that the optical flow estimator was trained on during inference. In our implementation, we use PWC-Net [57] to estimate the optical flow and fine-tune it on input patches of size 256×256 with a relatively small inter-frame motion magnitude. This optical flow estimator is expected to perform poorly on out-

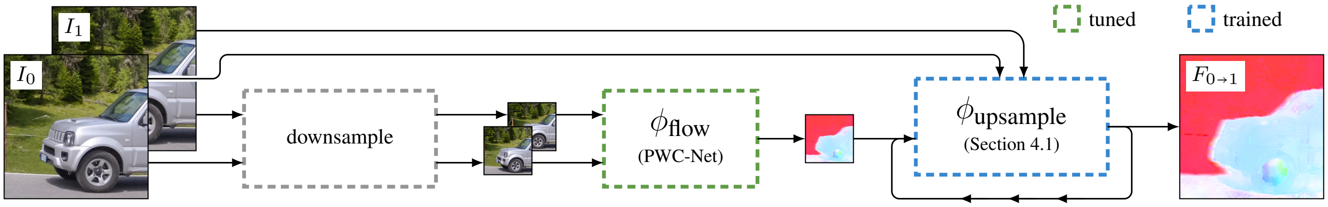


Figure 11. Overview of our iterative flow upsampling. Given two input images at a high resolution, we downsample them and then estimate the optical flow at a lower resolution. Our splatting-based synthesis requires full-resolution flow though, which is why we iteratively upsample the estimated flow guided by the input images. That is, the more we downsampled the more upsampling iterations we do.

	Middlebury Baker *et al.* [1]		Vimeo-90k Xue *et al.* [65]		Xiph-1K (4K scaled to 1K)		Xiph-2K (4K scaled to 2K)		Xiph-4K (from xiph.org)		runtime (seconds on a V100)		
	PSNR ↑	absolute rank	PSNR ↑	absolute rank	PSNR ↑	absolute rank	PSNR ↑	absolute rank	PSNR ↑	absolute rank	at 1K ↓	at 2K ↓	at 4K ↓
Ours w/o upsampling	<u>36.63</u>	1st of 3	<u>35.00</u>	1st of 3	<u>36.75</u>	1st of 3	<u>35.95</u>	1st of 3	33.93	3rd of 3	0.043	0.148	0.589
Ours at 1/2 res. w/ 2× upsampling	34.79	2nd of 3	33.89	2nd of 3	35.37	2nd of 3	35.52	2nd of 3	<u>34.68</u>	1st of 3	0.024	0.061	0.226
Ours at 1/4 res. w/ 4× upsampling	33.68	3rd of 3	32.82	3rd of 3	34.04	3rd of 3	34.81	3rd of 3	34.51	2nd of 3	0.023	0.041	0.137

Table 3. Evaluating the effect of flow upsampling on the interpolation quality and the runtime. Counterintuitively, estimating the motion on a lower resolution is not only beneficial in terms of runtime, but sometimes also quality (see 1/2 res. w/ 2× upsampling on Xiph-4K).

	Xiph-2K (4K scaled to 2K)		Xiph-4K (from xiph.org)	
	PSNR ↑	relative change	PSNR ↑	relative change
at 1/2 res. w/ bilinear up.	34.91	–	34.51	–
at 1/2 res. w/ our up.	<u>35.52</u>	+ 0.61 dB	<u>34.68</u>	+ 0.17 dB
at 1/4 res. w/ bilinear up.	32.10	–	33.10	–
at 1/4 res. w/ our up.	34.81	+ 2.71 dB	34.51	+ 1.41 dB

Table 4. Comparison of our iterative flow upsampling with a baseline that only uses bilinear interpolation to upsample the flow.

of-domain high-resolution footage such as 4K inputs. But by downsampling our inputs to resemble the data on which the flow estimation network was trained, we achiever better interpolation result at high resolutions (see Table 3).

4.2. Baseline Comparison

We compare our proposed iterative flow upsampling to a baseline that only uses bilinear interpolation to upsample the flow in Figure 4, which shows that it is key to upsample the flow in a guided manner. Without a ϕ_{upsample} trained specifically for this task, the drop in interpolation quality, especially when estimating the motion at 1/4 resolution and then upsampling it by a $4\times$, would be too severe to usefully benefit from the improved computational efficiency.

5. Stable Softmax Splatting

The challenge with splatting is that multiple pixels from the source image can map to the same location in the target, which creates an ambiguity that in the context of deep learning needs to be resolved differentiably. Softmax splatting is a recent solution to this problem [43], which has already found many applications [16, 17, 23, 32, 69]. However, the way softmax splatting is implemented is not numerically stable, which we subsequently outline and address.

Given an image I_0, an optical flow $F_{0\to t}$ that maps pixels in I_0 to the target time t and a weight map Z_0 to resolve ambiguities where multiple pixels from I_0 map to the same target location, softmax splatting $\overrightarrow{\sigma}$ is defined as:

$$\overrightarrow{\sigma}\left(I_0, F_{0\to t}, Z_0\right) = \frac{\overrightarrow{\Sigma}\left(\exp(Z_0) \cdot I_0, F_{0\to t}\right)}{\overrightarrow{\Sigma}\left(\exp(Z_0), F_{0\to t}\right)} \tag{6}$$

where $\overrightarrow{\Sigma}(\cdot)$ is summation splatting [43] and Z_0 can be thought of as an importance metric that acts like a soft inverse z-buffer (a hard z-buffer is not differentiable [41]).

The softmax operator is usually not implemented as defined since it is numerically unstable, $\exp(X)$ quickly exceeds 32-bit floating point when $X > 50$. Fortunately, since $\text{softmax}(X + c) = \text{softmax}(X)$ for any c, we can instead use $\text{softmax}(X')$ where $X' = X - \max(X)$ [18]. However, one cannot directly use this trick to numerically stabilize softmax splatting. Consider a weight map Z_0 with one element set to 1000 and all others $\in [0, 1]$. Shifting the weights by -1000 effectively sets all but one weight to 0 which then reduces the operation to average splatting, ignoring Z_0.

The weights must be shifted adaptively at the destination where multiple source pixels overlap. As such, we first warp Z_0 to time t as $Z_{0\to t}^{\max}$ which denotes the maximum weight for each pixel in the destination. This can be efficiently computed in parallel using an atomic max. Note that this step is and need not be differentiable as it is only used to make softmax splatting numerically stable. We can then subtract $Z_{0\to t}^{\max}[\boldsymbol{p}]$ from $Z_0[\boldsymbol{q}]$ before applying the exponential function when warping from a point $\boldsymbol{q}$ to $\boldsymbol{p}$, analogous to

	Middlebury Baker *et al.* [1]		Vimeo-90k Xue *et al.* [65]		Xiph-1K (4K scaled to 1K)		Xiph-2K (4K scaled to 2K)		Xiph-4K (from xiph.org)	
	PSNR ↑	relative change	PSNR ↑	relative change	PSNR ↑	relative change	PSNR ↑	relative change	PSNR ↑	relative change
original SoftSplat [43]	38.42	–	36.10	–	37.96	–	36.62	–	34.20	–
with our stable softmax splatting	38.59	+ 0.17 dB	36.18	+ 0.08 dB	37.99	+ 0.03 dB	36.74	+ 0.12 dB	34.62	+ 0.42 dB

Table 5. Our stable softmax splatting formulation leads to subtle but consistent improvements when applied to the original SoftSplat [43].

	XTEST-1K (4K scaled to 1K)		XTEST-2K (4K scaled to 2K)		XTEST-4K Sim *et al.* [56]	
	PSNR ↑	absolute rank	PSNR ↑	absolute rank	PSNR ↑	absolute rank
SepConv [45]	30.35	9th of 16	26.60	11th of 16	24.32	9th of 16
CtxSyn [42]	31.92	6th of 16	29.12	6th of 16	25.46	4th of 16
DAIN [2]	32.51	3rd of 16	31.49	2nd of 16	–	–
CAIN [10]	30.23	11th of 16	26.72	10th of 16	24.50	6th of 16
EDSC$_s$ [4]	30.54	8th of 16	26.37	12th of 16	–	–
EDSC$_m$ [4]	29.62	14th of 16	27.45	8th of 16	–	–
AdaCoF [30]	28.69	15th of 16	26.20	13th of 16	24.36	7th of 16
SoftSplat [43]	33.42	1st of 16	29.73	5th of 16	25.48	3rd of 16
BMBC [49]	30.04	12th of 16	25.46	15th of 16	–	–
RIFE [25]	32.32	4th of 16	27.49	7th of 16	24.67	5th of 16
SepConv++ [46]	29.78	13th of 16	26.12	14th of 16	24.36	7th of 16
CDFI [13]	30.30	10th of 16	26.89	9th of 16	–	–
XVFI [56]	31.54	7th of 16	31.12	3rd of 16	30.12	2nd of 16
XVFI$_v$ [56]	26.91	16th of 16	24.49	16th of 16	22.83	10th of 16
ABME [50]	32.08	5th of 16	30.15	4th of 16	–	–
Ours	33.31	2nd of 16	32.27	1st of 16	31.34	1st of 16

Table 6. Evaluating the $8\times$ interpolation capability of our approach in comparison to various other frame interpolation techniques on the XTEST [56] benchmark. Our approach generates better results and is an order of magnitude faster at doing so (see Figure 2).

what is typically done when implementing softmax. We thus define our numerically stable softmax splatting as:

$$\text{let } \boldsymbol{u} = \boldsymbol{p} - \big(\boldsymbol{q} + F_{0\to t}[\boldsymbol{q}]\big) \tag{7}$$

$$I_t[\boldsymbol{p}] = \frac{\sum_{\forall \boldsymbol{q} \in I_0} b(\boldsymbol{u}) \cdot \exp\big(Z_0[\boldsymbol{q}] - Z^{\max}_{0\to t}[\boldsymbol{p}]\big) \cdot I_0[\boldsymbol{q}]}{\sum_{\forall \boldsymbol{q} \in I_0} b(\boldsymbol{u}) \cdot \exp\big(Z_0[\boldsymbol{q}] - Z^{\max}_{0\to t}[\boldsymbol{p}]\big)} \tag{8}$$

$$b(\boldsymbol{u}) = \max(0, 1 - |\boldsymbol{u}_x|) \cdot \max(0, 1 - |\boldsymbol{u}_y|). \tag{9}$$

where $b(\cdot)$ is a bilinear kernel. Next, we demonstrate the benefits of this numerically stable softmax splatting operator on the task of frame interpolation. To do so, we reimplemented SoftSplat [43] but used our numerically stable softmax splatting instead of the official implementation. As shown in Table 5, the enhanced numerical stability of our implementation translates to subtle but consistent improvements in the interpolation quality. We expect similar improvements in other application domains such as in rolling shutter correction, video compression, video prediction, image animation, and various other synthesis tasks [16, 17, 23, 32, 69].

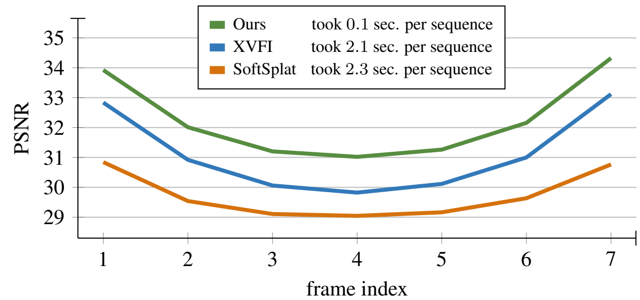


Figure 12. Evaluating the per-frame synthesis quality when performing $8\times$ interpolation on the XTEST-2K [56] benchmark.

6. Experiments

We subsequently provide additional implementation details, compare our splatting-based synthesis for frame interpolation to other approaches, and discuss its limitations.

6.1. Implementation

We use PWC-Net [57] trained on FlyingChairs [14] as the basis for the underlying optical flow estimator ϕ_{flow}. We fine-tune this flow estimator together with the seven parameters of the metrics extractor ϕ_{metrics} on the task of frame interpolation (Equation 1) with a Laplacian loss [42] using crops of size 256×256 from Vimeo-90k [65]. After convergence, we keep ϕ_{flow} and ϕ_{metrics} fixed while instead only training the iterative flow upsampling network ϕ_{upsample}, again using crops from the Vimeo-90k dataset. However, this time we uniformly sample the crop width from $\mathcal{U}(192, 448)$ and the crop height from $\mathcal{U}(192, 256)$ such that the upsampling network is supervised on various aspect ratios. During training, we run ϕ_{upsample} randomly for either one or two iterations.

6.2. Quantitative Evaluation

One of the benefits of our splatting-based synthesis is that once the motion has been estimated, interpolating frames only takes a few milliseconds. This makes our technique particularly useful for multi-frame interpolation, which we evaluate using the XTEST [56] benchmark. Since we have found the inter-frame motion in this benchmark to be rather extreme as its name suggests, we use our proposed approach with iterative $2\times$ down/upsampling on 2K inputs while using iterative $4\times$ down/upsampling on 4K inputs. The results of this experiment are shown in Table 6 and Figure 12. Aside

		Middlebury Baker *et al.* [1]		Vimeo-90k Xue *et al.* [65]		Xiph-1K (4K scaled to 1K)		Xiph-2K (4K scaled to 2K)		Xiph-4K (from xiph.org)		runtime (seconds on a V100)		
	venue	PSNR ↑	absolute rank	PSNR ↑	absolute rank	PSNR ↑	absolute rank	PSNR ↑	absolute rank	PSNR ↑	absolute rank	at 1K ↓	at 2K ↓	at 4K ↓
SepConv [45]	ICCV 2017	35.73	12th of 16	33.80	14th of 16	36.22	13th of 16	34.77	13th of 16	32.42	7th of 16	0.096	0.321	1.245
CtxSyn [42]	CVPR 2018	36.93	7th of 16	34.39	12th of 16	36.87	5th of 16	35.71	6th of 16	33.85	4th of 16	0.111	0.438	1.805
DAIN [2]	CVPR 2019	36.69	10th of 16	34.70	10th of 16	36.78	7th of 16	35.93	5th of 16	–	–	1.273	5.679	–
CAIN [10]	AAAI 2020	35.11	14th of 16	34.65	11th of 16	36.21	14th of 16	35.18	9th of 16	32.68	6th of 16	0.047	0.158	0.597
EDSC_s [4]	arXiv 2020	36.82	8th of 16	34.83	8th of 16	36.73	9th of 16	34.81	12th of 16	–	–	0.961	1.334	–
EDSC_m [4]	arXiv 2020	34.37	15th of 16	33.34	15th of 16	35.29	15th of 16	34.62	16th of 16	–	–	0.961	1.334	–
AdaCoF [30]	CVPR 2020	35.72	13th of 16	34.35	13th of 16	36.26	12th of 16	34.82	11th of 16	32.12	9th of 16	0.033	0.125	0.499
SoftSplat [43]	CVPR 2020	38.42	1st of 16	36.10	2nd of 16	37.96	1st of 16	36.62	1st of 16	34.20	2nd of 16	0.117	0.444	1.768
BMBC [49]	ECCV 2020	36.79	9th of 16	35.06	6th of 16	36.59	10th of 16	34.67	15th of 16	–	–	1.139	4.398	–
RIFE [25]	arXiv 2020	37.30	3rd of 16	35.61	3rd of 16	37.38	2nd of 16	36.16	3rd of 16	33.47	5th of 16	0.017	0.058	0.317
SepConv++ [46]	WACV 2021	37.28	4th of 16	34.83	8th of 16	36.83	6th of 16	34.84	10th of 16	32.23	8th of 16	0.092	0.364	1.455
CDFI [13]	CVPR 2021	37.14	5th of 16	35.17	4th of 16	37.05	3rd of 16	35.46	7th of 16	–	–	0.230	0.916	–
XVFI [56]	ICCV 2021	33.27	16th of 16	32.49	16th of 16	34.54	16th of 16	34.76	14th of 16	33.99	3rd of 16	0.114	0.297	0.964
XVFI_v [56]	ICCV 2021	37.09	6th of 16	35.07	5th of 16	36.98	4th of 16	35.19	8th of 16	32.12	9th of 16	0.114	0.297	0.964
ABME [50]	ICCV 2021	37.64	2nd of 16	36.18	1st of 16	36.53	11th of 16	36.50	2nd of 16	–	–	0.336	1.057	–
Ours	N/A	36.63	11th of 16	35.00	7th of 16	36.75	8th of 16	35.95	4th of 16	34.68	1st of 16	0.044	0.149	0.226

Table 7. Quantitative comparison of our proposed approach with various recent frame interpolation techniques that operate on two input images. The higher the resolution the better our approach ranks, and it performs best on the Xiph-4K test where it is also the fastest.

from being highly efficient when generating multiple frames between two given ones, our approach performs particularly well on XTEST which we attribute to its favorable ability to handle large motion. Further, the per-frame analysis shows that our splatting-based synthesis is temporally consistent.

We further evaluate our approach on common benchmark datasets as done in [46]. For this experiment, we use our interpolation pipeline without iterative flow upsampling on inputs of up to 2K and with $2\times$ down/upsampling for 4K inputs. As shown in Table 7, the higher the resolution the better our approach ranks and it performs best on Xiph-4K where it is also the fastest. While our approach does not yield state-of-the-art performance on low resolutions like with the Vimeo-90k test split, it is nevertheless surprising that it still outperforms both CtxSyn [42] and DAIN [2] on such small resolutions. After all, these methods not only splat the input images but also various feature representations before employing a synthesis network to generate the result which makes them much slower. In contrast, our synthesis is purely based on splatting without any subsequent refinement.

6.3. Qualitative Evaluation

Video frame interpolation results are best viewed as a motion picture, which is why we limit the qualitative evaluation in our main paper to only a single example in Figure 3 and kindly refer to our supplementary for more results.

6.4. Limitations

While generating results with our splatting-based synthesis is fast, it is wholly relying on the quality of the underlying optical flow estimate. In contrast, the refinement network that is used in related approaches that splat features before synthesizing the output using the warped features is able to account for minor inaccuracies in the estimated motion. Similarly, our splatting-based synthesis requires all the information that is necessary to interpolate the intermediate frame to be present in the input. However, this may not always be the case due to occlusions. In contrast, approaches with a refinement network can hallucinate missing content.

Furthermore, a synthesis approach like ours that solely relies on splatting will never be able to surpass an equivalent version that also utilizes a subsequent refinement network. As such, while our computational efficiency is unmatched, we consider the quantitative performance of our proposed interpolation pipeline as "good" but not "state-of-the-art" at low resolutions. The only reason we are able to claim state-of-the-art results at high resolutions is due to our iterative upsampling, but other methods could equally make use of this technique to improve their results at high resolutions.

7. Conclusion

In this paper, we show how to perform video frame interpolation while synthesizing the output solely through splatting. As such, synthesizing a frame only takes a few milliseconds once the inter-frame motion has been estimated, which makes our approach particularly useful for multi-frame interpolation. Furthermore, we combine this splatting-based synthesis approach with an iterative flow upsampling scheme which not only benefits the computational efficiency but also improves the interpolation quality at high resolutions.

References

[1] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A Database and Evaluation Methodology for Optical Flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[2] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-Aware Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[3] Tim Brooks and Jonathan T. Barron. Learning to Synthesize Motion Blur. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[4] Xianhang Cheng and Zhenzhong Chen. Multiple Video Frame Interpolation via Enhanced Deformable Separable Convolution. *arXiv/2006.08070*, 2020.

[5] Xianhang Cheng and Zhenzhong Chen. Video Frame Interpolation via Deformable Separable Convolution. In *AAAI Conference on Artificial Intelligence*, 2020.

[6] Zhixiang Chi, Rasoul Mohammadi Nasiri, Zheng Liu, Juwei Lu, Jin Tang, and Konstantinos N. Plataniotis. All at Once: Temporally Adaptive Multi-Frame Interpolation With Advanced Motion Modeling. In *European Conference on Computer Vision*, 2020.

[7] Byeong-Doo Choi, Jong-Woo Han, Chang-Su Kim, and Sung-Jea Ko. Motion-Compensated Frame Interpolation Using Bilateral Motion Estimation and Adaptive Overlapped Block Motion Compensation. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(4):407–416, 2007.

[8] Byung-Tae Choi, Sung-Hee Lee, and Sung-Jea Ko. New Frame Rate Up-Conversion Using Bi-Directional Motion Estimation. *IEEE Transactions on Consumer Electronics*, 46(3):603–609, 2000.

[9] Myungsub Choi, Janghoon Choi, Sungyong Baik, Tae Hyun Kim, and Kyoung Mu Lee. Scene-Adaptive Video Frame Interpolation via Meta-Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[10] Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. Channel Attention Is All You Need for Video Frame Interpolation. In *AAAI Conference on Artificial Intelligence*, 2020.

[11] Myungsub Choi, Suyoung Lee, Heewon Kim, and Kyoung Mu Lee. Motion-Aware Dynamic Architecture for Efficient Frame Interpolation. In *IEEE International Conference on Computer Vision*, 2021.

[12] Salih Dikbas and Yucel Altunbasak. Novel True-Motion Estimation Algorithm and Its Application to Motion-Compensated Temporal Frame Interpolation. *IEEE Transactions on Image Processing*, 22(8):2931–2945, 2013.

[13] Tianyu Ding, Luming Liang, Zhihui Zhu, and Ilya Zharkov. CDFI: Compression-Driven Network Design for Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[14] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow With Convolutional Networks. In *IEEE International Conference on Computer Vision*, 2015.

[15] Abdelrahman Eldesokey and Michael Felsberg. Normalized Convolution Upsampling for Refined Optical Flow Estimation. *arXiv/2102.06979*, 2021.

[16] Bin Fan and Yuchao Dai. Inverting a Rolling Shutter Camera: Bring Rolling Shutter Images to High Framerate Global Shutter Video. In *IEEE International Conference on Computer Vision*, 2021.

[17] Runsen Feng, Zongyu Guo, Zhizheng Zhang, and Zhibo Chen. Versatile Learned Video Compression. *arXiv/2111.03386*, 2021.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[19] Shurui Gui, Chaoyue Wang, Qihua Chen, and Dacheng Tao. FeatureFlow: Robust Video Interpolation via Structure-to-Texture Generation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[20] Taehyeun Ha, Seongjoo Lee, and Jaeseok Kim. Motion Compensated Frame Interpolation by New Block-Based Motion Estimation Algorithm. *IEEE Transactions on Consumer Electronics*, 50(2):752–759, 2004.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep Into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *IEEE International Conference on Computer Vision*, 2015.

[22] Evan Herbst, Steve Seitz, and Simon Baker. Occlusion Reasoning for Temporal Interpolation Using Optical Flow. Technical report, 2009.

[23] Aleksander Holynski, Brian L. Curless, Steven M. Seitz, and Richard Szeliski. Animating Pictures With Eulerian Motion Fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[24] Ai-Mei Huang and Truong Q. Nguyen. Correlation-Based Motion Vector Processing With Adaptive Interpolation Scheme for Motion-Compensated Frame Interpolation. *IEEE Transactions on Image Processing*, 18(4):740–752, 2009.

[25] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. RIFE: Real-Time Intermediate Flow Estimation for Video Frame Interpolation. *arXiv/2011.06294*, 2020.

[26] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial Transformer Networks. In *Advances in Neural Information Processing Systems*, 2015.

[27] Seong-Gyun Jeong, Chul Lee, and Chang-Su Kim. Motion-Compensated Frame Interpolation Based on Multihypothesis Motion Estimation and Texture Optimization. *IEEE Transactions on Image Processing*, 22(11):4497–4509, 2013.

[28] Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran. FLAVR: Flow-Agnostic Video Representations for Fast Frame Interpolation. *arXiv/2012.08512*, 2020.

[29] Soo Ye Kim, Jihyong Oh, and Munchurl Kim. FISR: Deep Joint Frame Interpolation and Super-Resolution With a Multi-Scale Temporal Loss. In *AAAI Conference on Artificial Intelligence*, 2020.

[30] Hyeongmin Lee, Taeoh Kim, Tae-Young Chung, Daehyun Pak, Yuseok Ban, and Sangyoun Lee. AdaCoF: Adaptive Collaboration of Flows for Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[31] Siyao Li, Shiyu Zhao, Weijiang Yu, Wenxiu Sun, Dimitris N. Metaxas, Chen Change Loy, and Ziwei Liu. Deep Animation Video Interpolation in the Wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[32] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[33] Songnan Lin, Jiawei Zhang, Jinshan Pan, Zhe Jiang, Dongqing Zou, Yongtian Wang, Jing Chen, and Jimmy S. J. Ren. Learning Event-Driven Video Deblurring and Interpolation. In *European Conference on Computer Vision*, 2020.

[34] Yihao Liu, Liangbin Xie, Siyao Li, Wenxiu Sun, Yu Qiao, and Chao Dong. Enhanced Quadratic Video Interpolation. *arXiv/2009.04642*, 2020.

[35] Yu-Lun Liu, Yi-Tung Liao, Yen-Yu Lin, and Yung-Yu Chuang. Deep Video Frame Interpolation Using Cyclic Frame Generation. In *AAAI Conference on Artificial Intelligence*, 2019.

[36] Ziwei Liu, Raymond A. Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video Frame Synthesis Using Deep Voxel Flow. In *IEEE International Conference on Computer Vision*, 2017.

[37] Kunming Luo, Chuan Wang, Shuaicheng Liu, Haoqiang Fan, Jue Wang, and Jian Sun. UPFlow: Upsampling Pyramid for Unsupervised Optical Flow Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[38] Dhruv Mahajan, Fu-Chung Huang, Wojciech Matusik, Ravi Ramamoorthi, and Peter N. Belhumeur. Moving Gradients: A Path-Based Method for Plausible Image Interpolation. *ACM Transactions on Graphics*, 28(3):42:1–42:11, 2009.

[39] Simone Meyer, Victor Cornillère, Abdelaziz Djelouah, Christopher Schroers, and Markus H. Gross. Deep Video Color Propagation. In *British Machine Vision Conference*, 2018.

[40] Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus H. Gross, and Christopher Schroers. PhaseNet for Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[41] Thu Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yong-Liang Yang. RenderNet: A Deep Convolutional Network for Differentiable Rendering From 3D Shapes. In *Advances in Neural Information Processing Systems*, 2018.

[42] Simon Niklaus and Feng Liu. Context-Aware Synthesis for Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[43] Simon Niklaus and Feng Liu. Softmax Splatting for Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[44] Simon Niklaus, Long Mai, and Feng Liu. Video Frame Interpolation via Adaptive Convolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[45] Simon Niklaus, Long Mai, and Feng Liu. Video Frame Interpolation via Adaptive Separable Convolution. In *IEEE International Conference on Computer Vision*, 2017.

[46] Simon Niklaus, Long Mai, and Oliver Wang. Revisiting Adaptive Convolutions for Video Frame Interpolation. In *IEEE Winter Conference on Applications of Computer Vision*, 2021.

[47] Simon Niklaus, Xuaner Cecilia Zhang, Jonathan T. Barron, Neal Wadhwa, Rahul Garg, Feng Liu, and Tianfan Xue. Learned Dual-View Reflection Removal. In *IEEE Winter Conference on Applications of Computer Vision*, 2021.

[48] Avinash Paliwal and Nima Khademi Kalantari. Deep Slow Motion Video Reconstruction With Hybrid Imaging System. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(7):1557–1569, 2020.

[49] Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. BMBC: Bilateral Motion Estimation With Bilateral Cost Volume for Video Interpolation. In *European Conference on Computer Vision*, 2020.

[50] Junheum Park, Chul Lee, and Chang-Su Kim. Asymmetric Bilateral Motion Estimation for Video Frame Interpolation. In *IEEE International Conference on Computer Vision*, 2021.

[51] Tomer Peleg, Pablo Szekely, Doron Sabo, and Omry Sendik. IM-Net for High Resolution Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[52] Fitsum A. Reda, Deqing Sun, Aysegul Dundar, Mohammad Shoeybi, Guilin Liu, Kevin J. Shih, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Unsupervised Video Interpolation Using Cycle Consistency. In *IEEE International Conference on Computer Vision*, 2019.

[53] Jonathan Shade, Steven J. Gortler, Li wei He, and Richard Szeliski. Layered Depth Images. In *Conference on Computer Graphics and Interactive Techniques*, 1998.

[54] Wang Shen, Wenbo Bao, Guangtao Zhai, Li Chen, Xiongkuo Min, and Zhiyong Gao. Blurry Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[55] Zhihao Shi, Xiaohong Liu, Kangdi Shi, Linhui Dai, and Jun Chen. Video Interpolation via Generalized Deformable Convolution. *arXiv/2008.10680*, 2020.

[56] Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. XVFI: eXtreme Video Frame Interpolation. In *IEEE International Conference on Computer Vision*, 2021.

[57] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[58] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *European Conference on Computer Vision*, 2020.

[59] Stepan Tulyakov, Daniel Gehrig, Stamatios Georgoulis, Julius Erbach, Mathias Gehrig, Yuanyou Li, and Davide Scaramuzza. Time Lens: Event-Based Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[60] Yang Wang, Haibin Huang, Chuan Wang, Tong He, Jue Wang, and Minh Hoai. GIF2Video: Color Dequantization and Temporal Interpolation of GIF Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[61] Zihao W. Wang, Weixin Jiang, Kuan He, Boxin Shi, Aggelos K. Katsaggelos, and Oliver Cossairt. Event-Driven Video Frame Synthesis. In *ICCV Workshops*, 2019.

[62] Chao-Yuan Wu, Nayan Singhal, and Philipp Krähenbühl. Video Compression Through Image Interpolation. In *European Conference on Computer Vision*, 2018.

[63] Xiaoyu Xiang, Yapeng Tian, Yulun Zhang, Yun Fu, Jan P. Allebach, and Chenliang Xu. Zooming Slow-Mo: Fast and Accurate One-Stage Space-Time Video Super-Resolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[64] Xiangyu Xu, Li Si-Yao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. Quadratic Video Interpolation. In *Advances in Neural Information Processing Systems*, 2019.

[65] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T. Freeman. Video Enhancement With Task-Oriented Flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019.

[66] Zhiyang Yu, Yu Zhang, Deyuan Liu, Dongqing Zou, Xijun Chen, Yebin Liu, and Jimmy S. Ren. Training Weakly Supervised Video Frame Interpolation With Events. In *IEEE International Conference on Computer Vision*, 2021.

[67] Liangzhe Yuan, Yibo Chen, Hantian Liu, Tao Kong, and Jianbo Shi. Zoom-in-to-Check: Boosting Video Interpolation via Instance-Level Discrimination. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[68] Haoxian Zhang, Yang Zhao, and Ronggang Wang. A Flexible Recurrent Residual Pyramid Network for Video Frame Interpolation. In *European Conference on Computer Vision*, 2020.

[69] Lili Zhao, Zezhi Zhu, Xuhu Lin, Xuezhou Guo, Qian Yin, Wenyi Wang, and Jianwen Chen. RAI-Net: Range-Adaptive LiDAR Point Cloud Frame Interpolation Network. *arXiv/2106.00496*, 2021.

[70] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon A. J. Winder, and Richard Szeliski. High-Quality Video View Interpolation Using a Layered Representation. *ACM Transactions on Graphics*, 23(3):600–608, 2004.