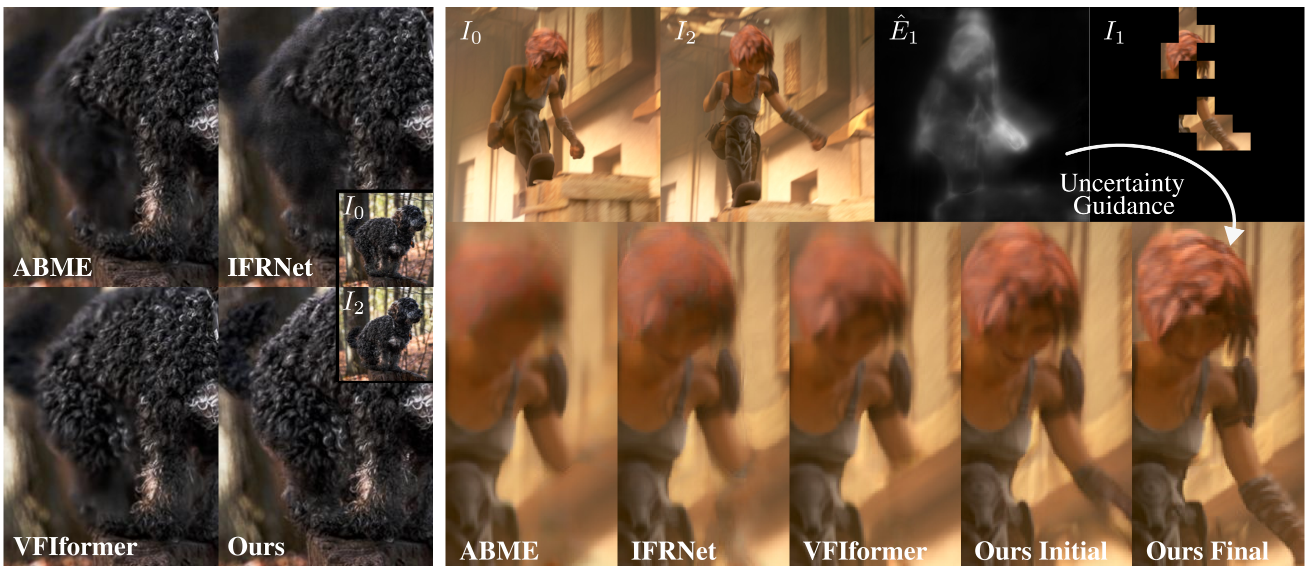
Frame Interpolation Transformer and Uncertainty Guidance

Markus Plack[1]* Karlis Martins Briedis[2,3] Abdelaziz Djelouah[3]
Matthias B. Hullin[1] Markus Gross[2,3] Christopher Schroers[3]

[1]University of Bonn, Bonn, Germany
[2]Department of Computer Science, ETH Zürich, Switzerland
[3]DisneyResearch|Studios, Zürich, Switzerland

mplack@cs.uni-bonn.de, karlis.briedis@inf.ethz.ch



Live Action Content Rendered Content

Figure 1. Our method achieves state-of-the-art results for frame interpolation. It produces sharp textures as highlighted on both live action (left) and rendered (right [15]) content. In addition to the interpolated frame, we estimate error maps that are helpful for quality checks in video production tools. More importantly, for rendered content it can be used to determine a subset of patches to render for the middle frame, which are then leveraged by our model to achieve production quality level results for a fraction of the rendering cost.

Abstract

Video frame interpolation has seen important progress in recent years, thanks to developments in several directions. Some works leverage better optical flow methods with improved splatting strategies or additional cues from depth, while others have investigated alternative approaches through direct predictions or transformers. Still, the problem remains unsolved in more challenging conditions such as complex lighting or large motion.

In this work, we are bridging the gap towards video production with a novel transformer-based interpolation network architecture capable of estimating the expected error together with the interpolated frame. This offers several advantages that are of key importance for frame interpolation usage: First, we obtained improved visual quality over several datasets. The improvement in terms of quality is also clearly demonstrated through a user study. Second, our method estimates error maps for the interpolated frame, which are essential for real-life applications on longer video sequences where problematic frames need to be flagged. Finally, for rendered content a partial rendering pass of the intermediate frame, guided by the predicted error, can be utilized during the interpolation to generate a new frame of superior quality. Through this error estimation, our method can produce even higher-quality intermediate frames using only a fraction of the time compared to a full rendering.

*Work done during an internship at DisneyResearch|Studios

1. Introduction

Video frame interpolation (VFI) is a classical video processing problem where the aim is to restore an intermediate frame in a given video sequence. This temporal inbetweening enables many practical applications, such as video editing [38], novel-view synthesis [26], video retiming, and slow motion generation [25]. Recent advances in VFI methods [13,24,28,30,37,48,53,55] have been continuously improving the interpolation quality, but the problem remains open due to complex lighting effects and large motion that are ubiquitous in real-life videos and can introduce severe artifacts for the existing methods.

We propose a transformer-based VFI architecture that processes both source and target frames in a unified framework and compensates motion through a tightly integrated optical flow estimation and cross-backward warping. Our model improves over the current state-of-the-art as supported by our extensive quantitative experiments and a user study.

Besides the improvements in terms of results, our model also predicts the interpolation uncertainty similar to approaches for artifact detection [4, 49] and adaptive sampling [29, 60]. This is of key importance for usage in a production context, where working with long sequences requires a way to automatically identify problematic frames. Uncertainty estimation also benefits Computer Graphics (CG) applications, as we use it to determine which frame patches do not have sufficient quality and optionally mark them for rendering. Thanks to our novel transformer-based model, the rendered patches from the middle frame naturally fit in the same unified VFI framework, achieving high quality levels at the fraction of the cost of rendering the full middle frame. Our paradigm is more compatible with current production renderers than CG specialized VFI works [5, 21, 66] which require the generation of specific G-buffers for the keyframes and the intermediate frame.

In summary, our contributions are as follows.

- We introduce a novel motion-based VFI method, that treats input and target frames in the same manner through a transformer-based architecture using masks.
- Our model achieves state-of-the-art performance as shown both in quantitative experiments and a user study.
- We perform output's uncertainty estimation subtask, which can be particularly beneficial for rendered content to achieve even better quality results.

2. Related work

While classical approaches to frame interpolation relied on optical flow and image warping [2, 52, 62], they have been surpassed by learning-based methods. We start our discussion with a short review of *direct*, *phase* and *kernel* based prediction methods, before going into more details with approaches using *motion* or *transformers*.

Direct methods were proposed using purely convolutional architectures [27, 36] or combining channel attention with a deep residual network [13]. Alternatively, Meyer *et al*. [40] show a *phase-based method* based on the idea that phase-shifts can be used to represent motion, and later extended with a learning-based component [39].

Kernel-based methods, as originally introduced by Niklaus *et al*. [44], aim to predict kernels for all pixels that are applied in a convolutional layer. Offset prediction has been used [9, 30] to reduce the necessary kernel size to handle large motion, making those methods conceptually more similar to motion-based ones. Various other extensions have been proposed, including prediction of separable kernels [45, 46], time input for arbitrary frame interpolation [10], a multi-scale architecture including cost volumes [8], multi-stage networks [20], different backbones [16, 54], and improving performance [50].

Most *motion-based methods* build on the work of optical flow estimation methods [18, 57, 61]. Some methods use the estimated motion between the input frames to forward splat them [23, 42, 43], while others aim to find the flow from the intermediate frame to the reference frames, allowing for an easy backward warping, either by estimating the flows directly [24, 28, 47, 48, 53], through other means [3, 25, 31, 41, 55], or combine both forward and backward warping approaches [17]. While most methods assume linear motion between the keyframes, others estimate non-linear motion by using more than two input frames [12, 19, 33, 34, 63] or with a learned prior [48].

Various other approaches have been proposed to improve estimation of large motion by treating small and large motion with equal priority [53], dynamically adapting the flow estimation to the motion magnitude and image resolution [55], or better strategies for feature propagation [1]. We adopt equal motion treatment by extending the scale-agnostic feature extraction [53, 58]. Most recently, CG specific frame interpolation algorithms have been introduced for 2D animation [56] and 3D rendering [5].

Error estimation of the optical flow is used by Chi *et al*. [11] for specific treatment, proposing predefined fixed models for the various error levels. This is different from our method, that learns to predict perceptual and L_2-based error maps for final interpolation result.

With the introduction of the transformer [59] and its adaptation to vision tasks [22], several *transformer-based* frame interpolation approaches have been proposed. Liu *et al*. [35] use a transformer architecture that incorporates convolutions inside attention layers, but does not include any motion compensation. VFIformer [37] uses cross-scale

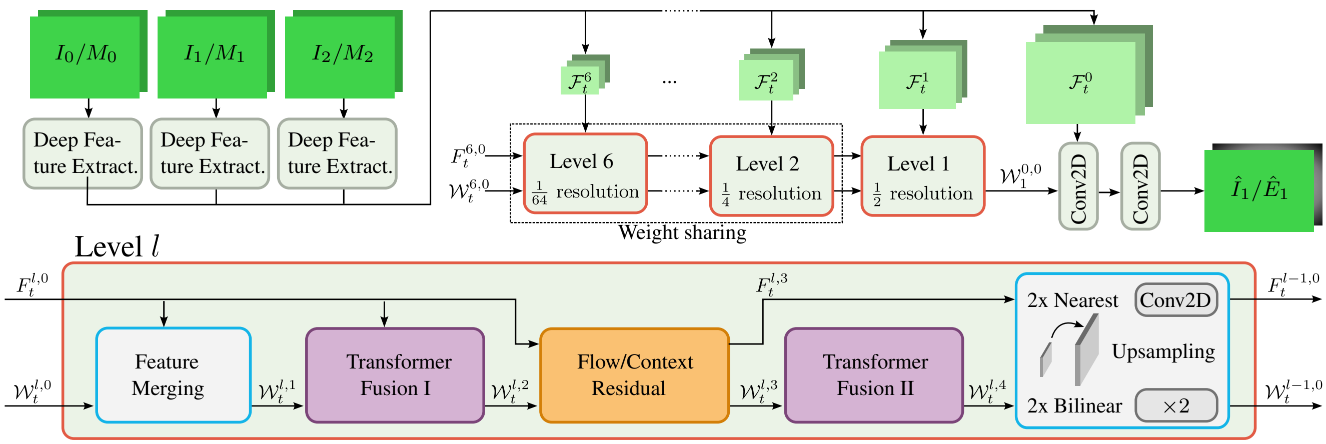


Figure 2. After extracting a feature pyramid $\{\mathcal{F}_t^l\}$ (**Deep Feature Extraction**) for each of the three frames (left) we pass a latent representation $\mathcal{W}_t$ along with a forward flow estimate F_t for each frame t through multiple levels of our reconstruction (center). At each level, after merging with the extracted features (**Feature Merging**), we update the latent representation using the initial flow estimate (**Transformer Fusion I**), followed by an update of the flow estimate and context vector from the new features (**Flow/Context Residual**) and another latent representation update using the new features and flows (**Transformer Fusion II**) before upsampling flow and features for the next level (**Upsampling**). Finally, we compute the interpolated Frame $\hat{I}_1$ and an estimate of the error $\hat{E}_1$ (top right).

window attention after warping the feature representations and TTVFI [32] uses an inconsistent region map inside a trajectory aware attention module. Both methods, however, cannot handle inputs of the middle frame and require an extra training of the upstream flow network, whereas our flow estimation is tightly integrated with the transformer fusion and trained end-to-end.

3. Method

The goal of our method is to interpolate two keyframes I_0, I_2 and find the intermediate frame $\hat{I}_1$ along with an estimate of the error $\hat{E}_1$. Subsequently, we analyze the error map and check if certain areas of the frame need to be rendered as we expect them to have insufficient quality. We then pass those additional masked inputs I_1 to the network along with the keyframes to get a final interpolated frame. Note that our method is well equipped to handle the common problem of two-frame interpolation without any changes to the architecture or training and that the additional inputs are entirely optional, *i.e*. we simply set $I_1 = 0$.

3.1. Interpolation network

Motivated by our goal to be able to handle arbitrary inputs, the overall architecture of our network is inspired by transformer architectures. This means that, opposed to common two-frame interpolation methods, there is little distinction within the network between the keyframes and the target frame. Instead, we equip each frame with a binary mask M_t indicating valid inputs to guide the interpolation. An overview of our method is given in Fig. 2.

We first extract a feature pyramid representation $\{\mathcal{F}_t^l\}_{l\in 0,\dots,6}$ for each of the inputs and process them in a coarse-to-fine manner with the same update blocks that share weights for the bottom 5 resolutions.

In each of the levels, we first merge the latent feature representations $\mathcal{W}_t^{l,i}$ with the respective input feature pyramid level. After that, they are updated in two *transformer fusion* blocks and a *flow/context residual* block in between that additionally updates the running flow estimates $F_t^{l,i}$, denoting the optical flow from t to $t+1$. Finally, the latent feature representations and flows are upsampled for processing in the next level.

In order to reduce the memory and compute costs, the processing of the topmost level is treated differently and consists of two convolutional layers.

Deep feature extraction. Our feature extraction is inspired by that of Reda et al. [53] to enable weight sharing on the lower levels of the reconstruction. We expand their idea by using a U-Net architecture instead of the original top-down approach. The reasoning behind this choice is that it more easily enables the network to capture semantically meaningful features on the upper levels of the pyramid without the need for many convolutional layers with large kernels or dilation.

First, we build image I_t^l and mask M_t^l pyramids, where image/mask l is downsampled by a factor of 2 to obtain level $l+1$. We concatenate both and pass them through a U-Net as illustrated in Fig. 3, keeping the last three layers as features. Finally, we concatenate all input and feature tensors of the same spatial resolution to build input feature pyramids $\{\mathcal{F}_t^l\}_{l\in 0,\dots,6}$ for $t \in \{0,1,2\}$. Note that all features from level two onward will be semantically similar

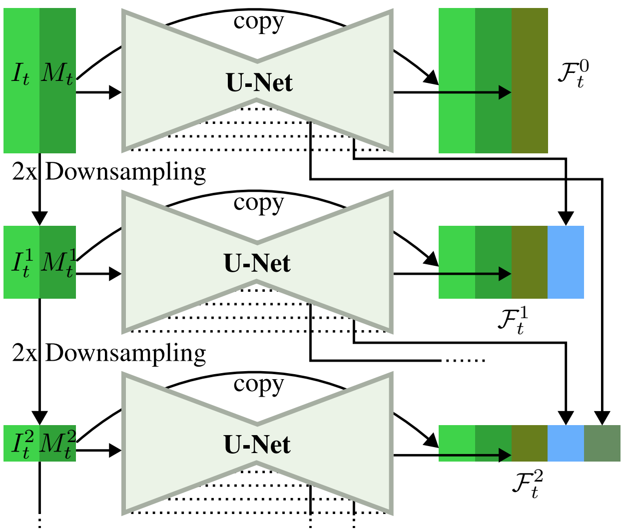


Figure 3. Illustration of our deep feature extraction module. The same U-Net is used to process the original inputs and all downsampled images/masks.

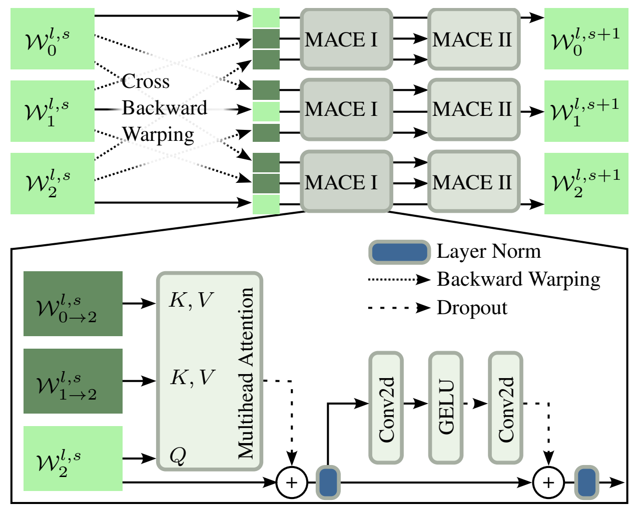


Figure 4. The transformer fusion module consists of two MACE blocks applied to all triplets after the cross backward warping.

and thus we can use weight sharing for all following modules on those levels.

Initialization and feature merging. On the lowest level we initialize the optical flows $F_t^{6,0}$ as 0 and set the latent feature representations $\mathcal{W}_t^{6,0}$ to a learned vector that is spatially repeated.

As the first step on each level, the upsampled pixel-wise features of the previous level, or the initial values, $\mathcal{W}_t^{l,0} \in \mathbb{R}^{D_l}$ are merged with their respective feature pyramid features $\mathcal{F}_t^l \in \mathbb{R}^{C_l}$, where $C_0 := 52$, $C_1 := 148$, $C_{i\in\{2..6\}} := 340$, and $D_l := C_l + 15$. Therefore, we only merge the first C_l channels of $\mathcal{W}_t^{l,0}$ with $\mathcal{F}_t^l$ while keeping the remaining 15 channels unaffected:

$$\mathcal{W}_t^{l,1} = \begin{bmatrix} M_t^l \mathcal{F}_t^l + (1 - M_t^l)\left[\mathcal{W}_t^{l,0}\right]_{0..C_l-1} \\ \left[\mathcal{W}_t^{l,0}\right]_{C_l..D_l-1} \end{bmatrix} \tag{1}$$

The purpose of the directly passed through channels is similar to explicit occlusion maps employed by other methods, but we leave the choice on how to best use those additional channels to be learned by the network.

Transformer fusion. To update the latent feature representation of each frame $t_0 \in \{0, 1, 2\}$, we use cross-backward warping to align the features of all other frames $t_i \neq t_0$ by rescaling the current flow estimate at stage s as

$$\mathcal{W}_{t_i \to t_0}^{l,s}(x, y) = \mathcal{W}_{t_i}^{l,s}((t_0 - t_i)F_{t_i}^{l,s}(x, y)) \tag{2}$$

for spatial indices (x, y) and using bilinear interpolation for non-integer coordinates. We treat $\mathcal{W}_{t_0}^{l,s}(x, y)$, $\mathcal{W}_{t_1\to t_0}^{l,s}(x, y)$, and $\mathcal{W}_{t_2\to t_0}^{l,s}(x, y)$ as tokens processed by the multihead attention module. Specifically, for each head i the per-pixel query, key and value tensors are computed as

$$Q_i = \mathbf{W}_i^Q \mathcal{W}_{t_0}^{l,s} \tag{3}$$

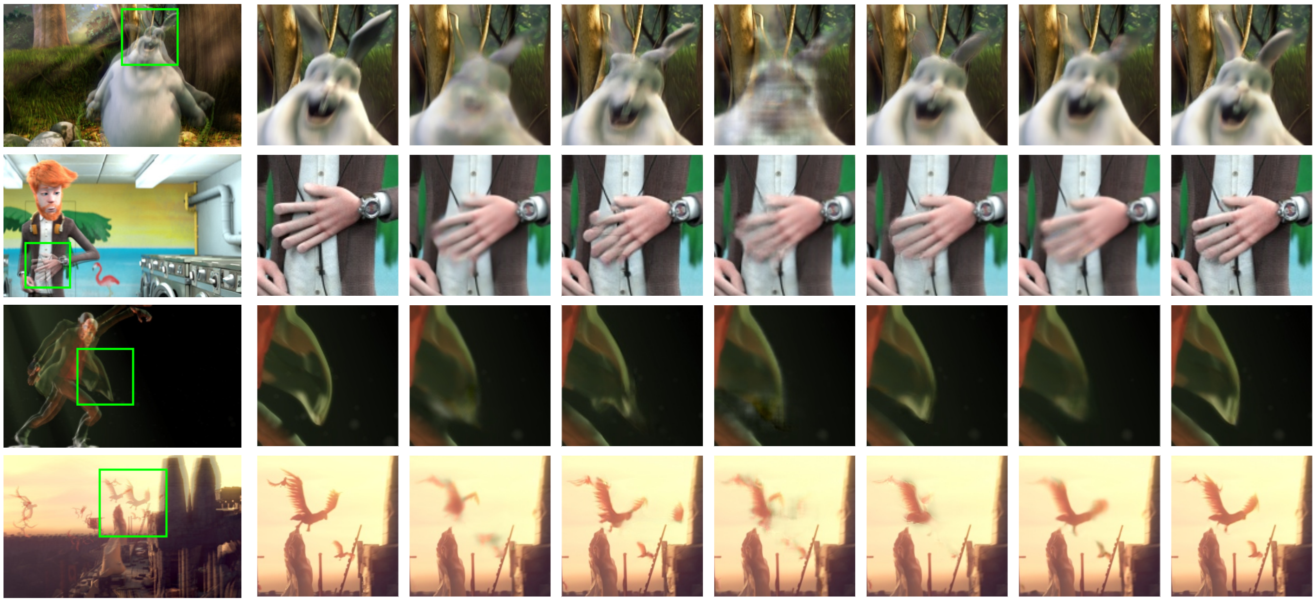
$$K_i = \mathbf{W}_i^K \left[\mathcal{W}_{t_1\to t_0}^{l,s}, \mathcal{W}_{t_2\to t_0}^{l,s}\right] \tag{4}$$

$$V_i = \mathbf{W}_i^V \left[\mathcal{W}_{t_1\to t_0}^{l,s}, \mathcal{W}_{t_2\to t_0}^{l,s}\right] \tag{5}$$

and the softmax of the query/key multiplication and the residual update from the weighted sum of the values are computed as in the original transformer [59].

Since our latent feature representations have an inherent spatial structure, we opt to replace the linear layers of the standard transformer with convolutional residual layers. We use two convolutions with kernel size 3, a dropout layer before and after the second convolution and a GELU activation after the first. In addition, we use layer normalization after the multihead attention and the convolutional layers, as is common in transformer architectures. We dub those modules **m**ultihead-**a**ttention **c**onvolutional **e**ncoders (MACE) and stack two of them for all transformer fusion modules as shown in Fig. 4 except for the second module on the second layer, which uses four MACE modules.

Flow residual. Initial tests suggested that a transformer module, as used for the feature updates, is a poor choice for updating the current flow estimate. Instead, we use a convolutional module for this task. After cross-backward warping the updated features to the reference frame, we pass each pair $(\mathcal{W}_t^{l,s}, \mathcal{W}_{v\to t}^{l,s})$ through a series of convolutions. The output contains the following tensors (stacked in channel dimension): Weight α_v, flow offset Δ_v^F, and context residual $\Delta_v^{\mathcal{W}}$ (We drop the level, time, and step indices of those



Inputs (overlaid) **GTruth** **ABME** **FILM L_S** **RIFE** **IFRNet** **VFIformer** **Ours L_S**

Figure 5. Visual comparison with other methods on rendered movie samples from [6, 7, 14, 15] using only keyframe inputs and no extra rendered patch.

for ease of notation). We apply softmax on the weights and update the flows and context features as

$$F_t^{l,3} = F_t^{l,2} + \frac{\sum_v e^{\alpha_v} \frac{1}{v-t} \Delta_v^F}{\sum_v e^{\alpha_v}} \tag{6}$$

$$\left[\mathcal{W}_t^{l,3}\right]_{C_l..D_l-1} = \left[\mathcal{W}_t^{l,2}\right]_{C_l..D_l-1} + \frac{\sum_v e^{\alpha_v} \Delta_v^{\mathcal{W}}}{\sum_v e^{\alpha_v}}. \tag{7}$$

Note how Δ_v^F needs to be rescaled to a forward flow for the update of $F_t^{l,3}$.

Miscellaneous. For the upsampling of the flows we use parameter-free bilinear interpolation by a scaling factor of two (Denoted by $\cdot_{\uparrow 2x}$) as

$$F_t^{l,0} = 2 F_t^{l+1,4}{}_{\uparrow 2x}. \tag{8}$$

The feature maps are passed through a resize convolution same as [53] to avoid checkerboard artifacts, *i.e.* a nearest-neighbor upsampling followed by a convolutional layer with kernel size 2 and D_l output feature channels.

For the final output, we pass the latent representations $\mathcal{W}_t^0$ together with the extracted features $\mathcal{F}_t^0$ through two convolutional layers with kernel sizes 3 and 1 respectively. The final output has five channels of which the first three form the color image $\hat{I}_t$ and the others correspond to the color error $\hat{E}_t^c$ and the perceptual error $\hat{E}_t^p$.

3.2. Uncertainty estimation

To train the error outputs $\hat{E}$ of the network we compute the target error maps as follows. Let I_t^{GT} be the ground truth frame at time t. We compute the error targets or 'ground truth' as

$$E_t^c = \|I_t^{GT} - \hat{I}_t\|_2 \tag{9}$$

where $\|\cdot\|_2$ denotes the $L2$ norm along the channel dimension. The perceptual error E_t^p follows the computation of LPIPS [65] without the spatial averaging. In order to prevent a detrimental influence of the error loss computations, we do not propagate gradients from the error map computations to the color output and only allow gradient flow to the error prediction of the network.

We want to use the error estimates $\hat{E}$ to find regions of the target frame that are expected to have insufficient quality, so we can render those areas and pass them to the network in a second pass to improve the quality. Assuming that most common renderers should be able to operate on a subset of rectangular tiles without a significant overhead, we average the error estimates for those tiles for which we chose a size of 16×16 pixels. Given a fixed budget for each frame, we simply select the tiles with the highest expected error and use them in the second interpolation pass.

3.3. Implementation and training

We follow common practice and train our network on triplets from the training set of Vimeo-90K [64]. Of the 51313 triplets of resolution 448×256 we set aside 802 for validation. For data augmentation we randomly crop windows of size 256, apply random spatial and temporal flipping and rotations in multiples of $90°$. We use empty middle frames for 50% of the training samples (*i.e.* $I_1 = 0$) and otherwise retain between $\frac{1}{480}$ and $\frac{1}{4}$ of 16×16 tiles as additional input (random at first and based on the predicted error for fine-tuning).

We train our L_1 variant for 2.1M iterations with batch size 4 using the Adam optimizer and $L1$ loss for the color output with weight 1.0 and for both error estimates with weight 0.01 each. We start with a learning rate of 5×10^{-5} and reduce it every 0.75M iterations by a factor of 0.464.

For our perceptual variant (L_S), we follow the same schedule, but add VGG and Style loss from [53] after 1.9M iterations, at which point we set the weights of the color, VGG and style loss as 10.0, 0.25 and 40. All losses are computed only for the center frame outputs, as we assume the keyframes are given and complete.

4. Experiments

We evaluate the performance of our method on the standard interpolation task (Sec. 4.1) and the efficiency of the uncertainty guidance (Sec. 4.2). We close with an ablation study (Sec. 4.3) and a discussion of limitations (Sec. 4.4).

Metrics. We measure our results using the common evaluation metrics peak signal-to-noise ratio (PSNR), structural similarity (SSIM) and the perceptual LPIPS [65]. In addition, we perform a user study for a qualitative evaluation.

Methods. We compare our method against ABME [48], AdaCoF [30], CAIN [13], FILM (L_1 and L_S) [53], IFRNet (Large) [28], RIFE [24], VFIformer [37], and XVFI [55].

Datasets. For the evaluation on traditional frame interpolation we use Vimeo90K [64], DAVIS [51], and SNU-FILM [13]. In addition, we evaluate on samples taken from the publicly available animated short films Big Buck Bunny [14], Cosmos Laundromat [7], Elephants Dream [6], and Sintel [15]. See supplementary material for more details and instructions to reproduce those datasets.

4.1. Traditional frame interpolation

We quantitatively evaluate our method on common datasets in Tab. 1 against the state of the art. Our L_1 variant shows the best PSNR and SSIM performance on all difficulty levels of SNU-FILM with a PSNR improvement of up to 0.21 dB in the hard category and a competitive performance on Vimeo90k and DAVIS. Our L_S version outperforms all others in terms of LPIPS on all datasets except DAVIS and demonstrates excellent PSNR and SSIM scores within its category. We show the performance on the animated short films in Tab. 2 where each variant outperforms all others within its category with respect to all metrics and on all datasets except Cosmos Laundromat, where both nevertheless yield good results.

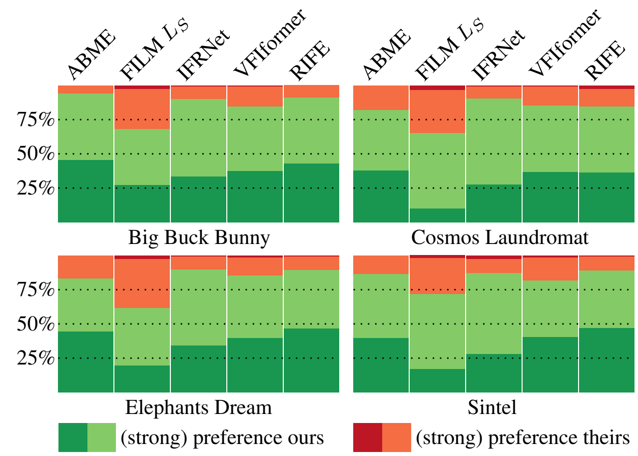


Figure 6. User study on the animated short film datasets. On average, users had a normal/strong preference for our method for 48/34% of all votes. For each of the short films, we use a representative subset of 30 samples and collected a total of 3158 AB comparisons from 69 participants, most of whom are computer graphics/vision students and graduates.

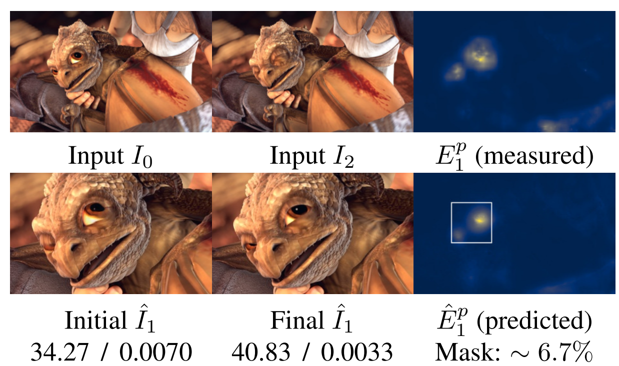


Figure 7. The closing of the eyes proves difficult to interpolate, but the expected perceptual error $\hat{E}_1^p$ closely matches the true error E_1^p. Passing the part of the middle frame indicated by the white box to the network we get a significantly improved interpolation. Numbers below are PSNR/LPIPS. Sample is from [15].

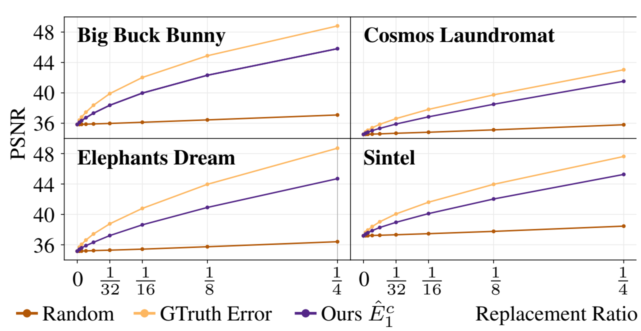


Figure 8. Replacement of tiles based on random sampling, highest ground truth error, *i.e.* the upper boundary of achievable PSNR, and our color error estimation $\hat{E}_1^c$.

Method		Vimeo90k			DAVIS			SNU-FILM Easy			SNU-FILM Medium			SNU-FILM Hard			SNU-FILM Extreme			Rank Count	
		PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	1st	2nd
ABME	'21	36.22	0.9808	0.0217	26.47	0.8601	0.1481	39.74	0.9904	0.0228	35.85	0.9792	0.0380	30.62	0.9367	0.0668	25.44	0.8642	0.1271	0	1
AdaCoF	'20	34.38	0.9717	0.0309	25.10	0.8221	0.1550	38.85	0.9902	0.0202	35.07	0.9757	0.0372	29.47	0.9246	0.0764	24.31	0.8442	0.1493	0	0
FILM L_1	'22	36.06	0.9804	0.0201	27.31	0.8784	**0.0846**	40.20	0.9909	0.0186	36.01	0.9795	**0.0321**	30.49	0.9359	0.0578	25.20	0.8601	**0.1071**	3	4
IFRNet	'22	36.20	0.9808	**0.0193**	27.46	0.8797	0.0926	40.10	0.9906	0.0210	36.12	0.9797	0.0328	30.63	0.9368	**0.0570**	25.26	0.8609	0.1138	2	1
RIFE	'22	35.61	0.9780	0.0227	26.70	0.8616	0.1126	40.06	0.9907	0.0188	35.72	0.9789	0.0325	30.09	0.9331	0.0665	24.84	0.8537	0.1395	0	0
VFIformer	'22	**36.50**	**0.9816**	0.0202	**27.60**	**0.8829**	0.0939	40.13	0.9907	0.0181	36.09	0.9799	0.0333	30.67	0.9378	0.0612	25.43	0.8643	0.1190	4	5
XVFI	'21	35.06	0.9758	0.0234	25.71	0.8409	0.1365	39.99	0.9905	**0.0177**	35.36	0.9779	0.0322	29.56	0.9271	0.0752	24.14	0.8446	0.1551	1	1
Ours L_1		36.34	0.9814	0.0204	27.46	0.8803	0.0923	**40.25**	**0.9909**	0.0202	**36.29**	**0.9803**	0.0344	**30.88**	**0.9386**	0.0604	**25.61**	**0.8655**	0.1130	8	6
CAIN	'20	34.67	0.9733	0.0311	26.03	0.8415	0.1787	39.96	0.9903	0.0204	35.64	0.9779	0.0385	29.91	0.9295	0.0898	24.78	0.8510	0.1803	0	0
FILM L_S	'22	35.87	0.9790	0.0132	27.00	0.8709	**0.0679**	**40.15**	**0.9906**	0.0121	35.90	0.9786	0.0215	30.33	0.9333	0.0434	25.07	0.8552	0.0899	3	15
Ours L_S		**36.08**	**0.9799**	**0.0126**	**27.03**	**0.8712**	0.0706	40.10	0.9905	**0.0118**	**36.07**	**0.9790**	**0.0209**	**30.61**	**0.9351**	**0.0420**	**25.35**	**0.8594**	**0.0864**	15	3

Table 1. Live action VFI results. We list perceptually trained methods separately below the other methods. All metrics were obtained by running the implementations provided by the authors.

Method		Big Buck Bunny			Cosmos Laundromat			Elephants Dream			Sintel			Rank #	
		PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	1st	2nd
ABME	'21	35.60	0.9790	0.0323	34.47	0.9400	0.0823	34.80	0.9647	0.0453	36.83	0.9673	0.0495	0	0
AdaCoF	'20	34.17	0.9740	0.0413	33.83	0.9328	0.0877	33.52	0.9551	0.0560	34.73	0.9550	0.0703	0	0
FILM L_1	'22	35.50	0.9795	0.0282	34.42	0.9397	0.0678	34.70	0.9652	0.0390	36.71	0.9672	0.0395	0	4
IFRNet	'22	35.46	0.9810	0.0292	34.25	0.9399	**0.0674**	34.58	0.9659	0.0419	36.27	0.9683	0.0462	1	0
RIFE	'22	35.05	0.9767	0.0354	34.32	0.9379	0.0808	34.54	0.9615	0.0484	36.33	0.9638	0.0521	0	0
VFIformer	'22	35.97	0.9811	0.0365	**34.56**	**0.9415**	0.0750	35.06	0.9675	0.0406	36.94	0.9694	0.0432	2	6
XVFI	'21	34.64	0.9757	0.0371	34.09	0.9356	0.0774	34.00	0.9595	0.0503	35.51	0.9605	0.0585	0	0
Ours L_1		**35.98**	**0.9815**	**0.0262**	34.55	0.9407	0.0762	**35.25**	**0.9680**	**0.0372**	**37.25**	**0.9697**	**0.0393**	9	2
CAIN	'20	33.38	0.9733	0.0414	33.92	**0.9369**	0.0982	33.57	0.9571	0.0577	35.18	0.9586	0.0727	1	0
FILM L_S	'22	35.31	0.9787	0.0239	**34.20**	0.9361	0.0389	34.67	0.9643	0.0314	36.65	0.9661	0.0316	1	11
Ours L_S		**35.73**	**0.9805**	**0.0218**	34.08	0.9348	**0.0347**	**35.05**	**0.9666**	**0.0295**	**37.01**	**0.9678**	**0.0302**	10	1

Table 2. Animated short film VFI results. We list perceptually trained methods separately below the other methods. All metrics were obtained by running the implementations provided by the authors. Only keyframes were used and no extra rendered patches.

To further support our claim that our method performs well in terms of visual quality, we conduct an extensive user study. We roughly follow the approach of [42] and asked users to compare methods side by side, but included an option for a strong preference. We show one sample of each film in Fig. 5 and give the results in Fig. 6. We refer to the supplementary material for more details and results.

4.2. Uncertainty guided interpolation

We will demonstrate the advantages of our uncertainty guidance in two experiments by analyzing the ability of our error prediction to select appropriate patches in the interpolated image first, and secondly showing the quality improvement by passing additional patches to the network.

In Fig. 8 we demonstrate the PSNR improvement when we use our error estimation to replace a fraction of $16{\times}16$ tiles of the interpolated output by the corresponding ground truth. For comparison, we show the effect of random replacement as a baseline and a replacement of the tiles with the highest measured error as the optimal strategy. Replacing a quarter of the tiles, we achieve a PSNR improvement between 6.99 and 9.98 dB, whereas random replacement yields at most 1.27 dB.

Next we want to study the effect of additional inputs on the network output in separation from the error prediction. Therefore, we select tiles based on the true error and pass them into the network. We also compute the metrics when simply replacing the tiles in the interpolated output for our own method as a baseline and a selection of others for comparison. We plot the results in Fig. 9 which show that the perceptual quality is improved beyond the baseline approach.

We give a visual example of the full uncertainty guidance approach in Fig. 7, which shows how the correct region with high error is identified and the interpolation is improved by the additional inputs and refer to the supplementary material for additional results.

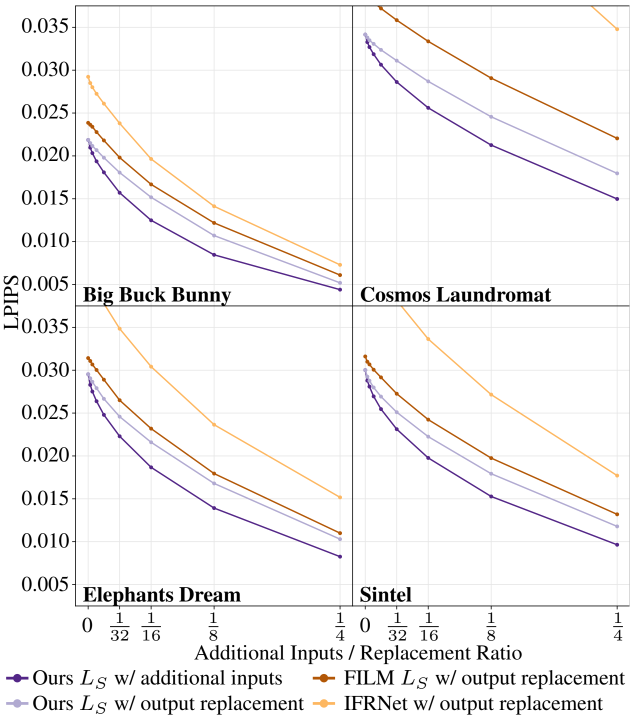


Figure 9. We show that the perceptual quality of the interpolation achieved by passing additional inputs to our method is better than the baseline approach of replacing the worst patches of the interpolation based on color error. For reference, we also show the curves when replacing the outputs of FILM L_S and IFRNet, the two follow up methods in terms of perceptual performance.

4.3. Ablation study

For an ablation study, we train different versions of our network to show the effect of the error estimation, the deep feature extraction and the shared frame processing. We use the same training procedure and color based loss for all variants as described in Sec. 3.3. The variants without error estimation differ only in the last convolutional layer (3 instead of 5 outputs) and do not use the error losses. The deep feature representation is replaced by the feature representation proposed by Reda et al. [53] and versions without shared frame processing only update the center frame in the transformer fusion and flow/context residual modules. The results are presented in Tab. 3 and highlight the advantages of the deep feature extraction and the shared frame processing for the interpolation quality.

4.4. Limitations

Very large motion or drastic visual changes can be missed by the error prediction and are hence not recovered through a second rendering pass. We show an example of this in the supplementary material. While the shared frame processing of the network through its transformer architecture should in theory be capable of recognizing missing objects that are unlikely to be occluded, we surmise that the current training dataset lacks sufficient examples to learn such behavior.

Lastly, the current network is relatively slow and big. *E.g.* VFIformer is on average 44.2% faster on Vimeo90k and needs about 27.6% fewer parameters. This makes training with more than two input frames challenging, even though the architecture supports it without any changes. We hope to improve this in the future, which could allow for better results through *e.g.* nonlinear flow estimates, or enable using our proposed architecture for other video processing tasks such as deblurring and super-resolution.

Error Est.	Deep Features	Shared Frames	Vimeo90k		Animated	
			PSNR	SSIM	PSNR	SSIM
✔	✔	✔	36.34	0.9814	35.75	0.9650
✔	✘	✔	36.28	0.9812	35.06	0.9633
✘	✔	✔	36.31	0.9813	35.71	0.9652
✘	✔	✘	35.82	0.9796	35.28	0.9634
✘	✘	✘	35.76	0.9793	35.14	0.9629

Table 3. Ablation study of our network design. We averaged the results of all animated films into a single score for each metric. We can see that the shared frame processing boosts the performance significantly, and the deep feature extraction adds a moderate improvement from the baseline, but is essential when interpolating animated content with the error estimation. The latter yields only a minor improvement, but its advantages demonstrated in Sec. 4.2 are significant.

5. Conclusion

In this work, we proposed a VFI method that incorporates optical flow motion compensation, deep feature extraction, error estimation, and shared frame processing in a transformer-based architecture. This enables our novel uncertainty-guided approach for animated content production, which can be used to greatly reduce the cost of rendering while maintaining a high visual quality as we have shown in our experiments. At the same time, our method achieves state-of-the-art results for traditional frame interpolation as demonstrated on multiple common benchmarks, and a superior visual quality confirmed by an extensive user study. Since our training procedure using masked inputs is similar to those of masked language models, a study of its properties remains an interesting direction for future work.

Acknowledgements. Markus Plack and Matthias B. Hullin are supported by the European Research Council under ERC Starting Grant “ECHO” (802192).

References

[1] Dawit Mureja Argaw and In So Kweon. Long-term video frame interpolation via feature propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3543–3552, June 2022. 2

[2] Simon Baker, Daniel Scharstein, JP 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

[3] 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. 2

[4] Mojtaba Bemana, Joachim Keinert, Karol Myszkowski, Michel Bätz, Matthias Ziegler, H-P Seidel, and Tobias Ritschel. Learning to predict image-based rendering artifacts with respect to a hidden reference image. In *Computer Graphics Forum*, volume 38, pages 579–589. Wiley Online Library, 2019. 2

[5] Karlis Martins Briedis, Abdelaziz Djelouah, Mark Meyer, Ian McGonigal, Markus Gross, and Christopher Schroers. Neural frame interpolation for rendered content. *ACM Transactions on Graphics (TOG)*, 40(6):1–13, 2021. 2

[6] (c) copyright 2006, Blender Foundation / Netherlands Media Art Institute / www.elephantsdream.org. Elephants dream, 2006. *Licensed under Creative Commons Attribution 2.5* (`https://creativecommons.org/licenses/by/2.5/`). 5, 6

[7] (CC) Blender Foundation | gooseberry.blender.org. Cosmos laundromat - first cycle - 2k, 2015. *Licensed under Creative Commons Attribution-ShareAlike 4.0* (`https://creativecommons.org/licenses/by-sa/4.0/`). 5, 6

[8] Zhiqi Chen, Ran Wang, Haojie Liu, and Yao Wang. PDWN: Pyramid deformable warping network for video interpolation. *IEEE Open Journal of Signal Processing*, 2:413–424, 2021. 2

[9] Xianhang Cheng and Zhenzhong Chen. Video frame interpolation via deformable separable convolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, Number 07, pages 10607–10614, 2020. 2

[10] Xianhang Cheng and Zhenzhong Chen. Multiple video frame interpolation via enhanced deformable separable convolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2

[11] Zhixiang Chi, Rasoul Mohammadi Nasiri, Zheng Liu, Yuanhao Yu, Juwei Lu, Jin Tang, and Konstantinos N Plataniotis. Error-aware spatial ensembles for video frame interpolation. *arXiv preprint arXiv:2207.12305*, 2022. 2

[12] Jinsoo Choi, Jaesik Park, and In So Kweon. High-quality frame interpolation via tridirectional inference. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 596–604, 2021. 2

[13] Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. Channel attention is all you need for video frame interpolation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, Number 07, pages 10663–10671, 2020. 2, 6

[14] Copyright (C) 2008 Blender Foundation — peach.blender.org. Big buck bunny, 2008. *Licensed under Creative Commons Attribution 3.0* (`http://creativecommons.org/licenses/by/3.0/`). 5, 6

[15] Copyright (c) Blender Foundation — durian.blender.org. Sintel, 2010. *Licensed under Creative Commons Attribution 3.0* (`http://creativecommons.org/licenses/by/3.0/`). 1, 5, 6

[16] Duolikun Danier, Fan Zhang, and David Bull. Enhancing deformable convolution based video frame interpolation with coarse-to-fine 3d cnn. *arXiv preprint arXiv:2202.07731*, 2022. 2

[17] Duolikun Danier, Fan Zhang, and David Bull. St-mfnet: A spatio-temporal multi-flow network for frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3521–3531, June 2022. 2

[18] 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*, pages 2758–2766, 2015. 2

[19] Saikat Dutta, Arulkumar Subramaniam, and Anurag Mittal. Non-linear motion estimation for video frame interpolation using space-time convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1726–1731, 2022. 2

[20] Shurui Gui, Chaoyue Wang, Qihua Chen, and Dacheng Tao. Featureflow: Robust video interpolation via structure-to-texture generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14004–14013, 2020. 2

[21] Jie Guo, Xihao Fu, Liqiang Lin, Hengjun Ma, Yanwen Guo, Shiqiu Liu, and Ling-Qi Yan. Extranet: Real-time extrapolated rendering for low-latency temporal supersampling. *ACM Trans. Graph.*, 40(6), dec 2021. 2

[22] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2

[23] Ping Hu, Simon Niklaus, Stan Sclaroff, and Kate Saenko. Many-to-many splatting for efficient video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2

[24] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. RIFE: Real-time intermediate flow estimation for video frame interpolation. *arXiv preprint arXiv:2011.06294*, 2021. 2, 6

[25] 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*, pages 9000–9008, 2018. 2

[26] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field

cameras. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)*, 35(6), 2016. 2

[27] Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran. Flavr: Flow-agnostic video representations for fast frame interpolation. *arxiv*, 2021. 2

[28] 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. *arXiv preprint arXiv:2205.14620*, 2022. 2, 6

[29] Alexandr Kuznetsov, Nima Khademi Kalantari, and Ravi Ramamoorthi. Deep adaptive sampling for low sample count rendering. In *Computer Graphics Forum*, volume 37, pages 35–44. Wiley Online Library, 2018. 2

[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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5316–5325, 2020. 2, 6

[31] Sungho Lee, Narae Choi, and Woong Il Choi. Enhanced correlation matching based video frame interpolation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2839–2847, 2022. 2

[32] Chengxu Liu, Huan Yang, Jianlong Fu, and Xueming Qian. Ttvfi: Learning trajectory-aware transformer for video frame interpolation. *arXiv preprint arXiv:2207.09048*, 2022. 3

[33] Meiqin Liu, Chenming Xu, Chao Yao, Chunyu Lin, and Yao Zhao. Jnmr: Joint non-linear motion regression for video frame interpolation. *arXiv preprint arXiv:2206.04231*, 2022. 2

[34] Yihao Liu, Liangbin Xie, Li Siyao, Wenxiu Sun, Yu Qiao, and Chao Dong. Enhanced quadratic video interpolation. In *European Conference on Computer Vision*, pages 41–56. Springer, 2020. 2

[35] Zhouyong Liu, Shun Luo, Wubin Li, Jingben Lu, Yufan Wu, Shilei Sun, Chunguo Li, and Luxi Yang. Convtransformer: A convolutional transformer network for video frame synthesis. *arXiv preprint arXiv:2011.10185*, 2020. 2

[36] Gucan Long, Laurent Kneip, Jose M Alvarez, Hongdong Li, Xiaohu Zhang, and Qifeng Yu. Learning image matching by simply watching video. In *European Conference on Computer Vision*, pages 434–450. Springer, 2016. 2

[37] 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*, pages 3532–3542, 2022. 2, 6

[38] Simone Meyer, Victor Cornillère, Abdelaziz Djelouah, Christopher Schroers, and Markus Gross. Deep video color propagation. In *Proceedings of the British Machine Vision Conference BMVC*, 2018. 2

[39] Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus Gross, and Christopher Schroers. Phasenet for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[40] Simone Meyer, Oliver Wang, Henning Zimmer, Max Grosse, and Alexander Sorkine-Hornung. Phase-based frame interpolation for video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1410–1418, 2015. 2

[41] Simon Niklaus, Ping Hu, and Jiawen Chen. Splatting-based synthesis for video frame interpolation. *arXiv preprint arXiv:2201.10075*, 2022. 2

[42] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1710, 2018. 2, 7

[43] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5437–5446, 2020. 2

[44] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 670–679, 2017. 2

[45] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 261–270, 2017. 2

[46] Simon Niklaus, Long Mai, and Oliver Wang. Revisiting adaptive convolutions for video frame interpolation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1099–1109, 2021. 2

[47] 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*, pages 109–125. Springer, 2020. 2

[48] 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*, pages 14539–14548, 2021. 2, 6

[49] Anjul Patney and Aaron Lefohn. Detecting aliasing artifacts in image sequences using deep neural networks. In *Proceedings of the Conference on High-Performance Graphics*, pages 1–4, 2018. 2

[50] Tomer Peleg, Pablo Szekely, Doron Sabo, and Omry Sendik. Im-net for high resolution video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2398–2407, 2019. 2

[51] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–732, 2016. 6

[52] Lars Lau Rakêt, Lars Roholm, Andrés Bruhn, and Joachim Weickert. Motion compensated frame interpolation with a symmetric optical flow constraint. In *International Symposium on Visual Computing*, pages 447–457. Springer, 2012. 2

[53] Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. FILM: Frame interpolation for large motion. In *European Conference on Computer Vision*, 2022. 2, 3, 5, 6, 8

[54] Zhihao Shi, Xiangyu Xu, Xiaohong Liu, Jun Chen, and Ming-Hsuan Yang. Video frame interpolation transformer. *arXiv preprint arXiv:2111.13817*, 2021. 2

[55] Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. XVFI: Extreme video frame interpolation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14489–14498, 2021. 2, 6

[56] Li Siyao, Shiyu Zhao, Weijiang Yu, Wenxiu Sun, Dimitris Metaxas, Chen Change Loy, and Ziwei Liu. Deep animation video interpolation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6587–6595, June 2021. 2

[57] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018. 2

[58] Marc Comino Trinidad, Ricardo Martin Brualla, Florian Kainz, and Janne Kontkanen. Multi-view image fusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4101–4110, 2019. 2

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017. 2, 4

[60] Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (TOG)*, 37(4):1–15, 2018. 2

[61] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392, 2013. 2

[62] Manuel Werlberger, Thomas Pock, Markus Unger, and Horst Bischof. Optical flow guided TV-L 1 video interpolation and restoration. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 273–286. Springer, 2011. 2

[63] Xiangyu Xu, Li Siyao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. Quadratic video interpolation. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[64] 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. 5, 6

[65] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5, 6

[66] Henning Zimmer, Fabrice Rousselle, Wenzel Jakob, Oliver Wang, David Adler, Wojciech Jarosz, Olga Sorkine-Hornung, and Alexander Sorkine-Hornung. Path-space motion estimation and decomposition for robust animation filtering. *Computer Graphics Forum (Proceedings of EGSR)*, 34(4), June 2015. 2