FILM: Frame Interpolation for Large Motion

Fitsum Reda[1], Janne Kontkanen[1], Eric Tabellion[1], Deqing Sun[1], Caroline Pantofaru[1], and Brian Curless[1,2]

[1] Google Research
[2] University of Washington

Abstract. We present a frame interpolation algorithm that synthesizes an engaging slow-motion video from near-duplicate photos which often exhibit large scene motion. Near-duplicates interpolation is an interesting new application, but large motion poses challenges to existing methods. To address this issue, we adapt a feature extractor that shares weights across the scales, and present a "scale-agnostic" motion estimator. It relies on the intuition that large motion at finer scales should be similar to small motion at coarser scales, which boosts the number of available pixels for large motion supervision. To inpaint wide disocclusions caused by large motion and synthesize crisp frames, we propose to optimize our network with the Gram matrix loss that measures the correlation difference between features. To simplify the training process, we further propose a unified single-network approach that removes the reliance on additional optical-flow or depth network and is trainable from frame triplets alone. Our approach outperforms state-of-the-art methods on the Xiph large motion benchmark while performing favorably on Vimeo-90K, Middlebury and UCF101. Source codes and pre-trained models are available at https://film-net.github.io.

Keywords: video synthesis, interpolation, optical flow, feature pyramid

1 Introduction

Frame interpolation – synthesizing intermediate images between a pair of input frames – is an important problem with increasing reach. It is often used for temporal up-sampling to increase refresh rate or create slow-motion videos.

Recently, a new use case has emerged. Digital photography, especially with the advent of smartphones, has made it effortless to take several pictures within a few seconds, and people naturally do so often in their quest for just the right photo that captures the moment. These "near duplicates" create an exciting opportunity: interpolating between them can lead to surprisingly engaging videos that reveal scene (and some camera) motion, often delivering an even more pleasing sense of the moment than any one of the original photos.

Unlike video, however, the temporal spacing between near duplicates can be a second or more, with commensurately large scene motion, posing a major challenge for existing interpolation methods. Frame interpolation between consecutive video frames, which often exhibit small motion, has been studied

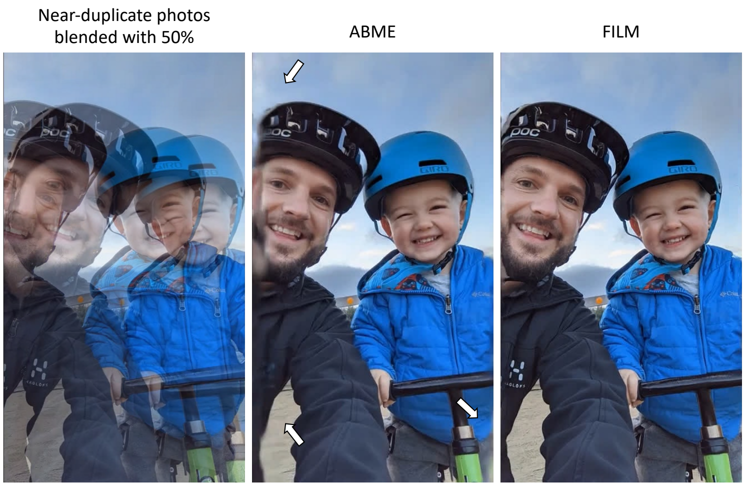


Fig. 1. Near-duplicate photos interpolation with ABME [23], showing large artifacts, and our FILM, showing improvements.

extensively, and recent methods [18,3,23,8] show impressive results for this scenario. However, little attention has been given to interpolation for large scene motion, commonly present in near duplicates. The work of [28] attempted to tackle the large motion problem by training on an extreme motion dataset, but its effectiveness is limited when tested on small motion [23].

In this work, we instead propose a network that generalizes well to both small and large motion. Specifically, we adapt a multi-scale feature extractor from [31] that shares weights across the scales and present a "scale-agnostic" bi-directional motion estimation module. Our approach relies on the intuition that large motion at finer scales should be similar to small motion at coarser scales, thus increasing the number of pixels (as finer scale is higher resolution) available for large motion supervision. We found this approach to be surprisingly effective in handling large motion by simply training on regular frames (see Figure 1).

We also observed that, while the state-of-the-art methods score well on benchmarks [16,2,34], the interpolated frames often appear blurry, especially in large disoccluded regions that arise from large motions. Here, we propose to optimize our models with the Gram matrix loss, which matches the *auto-correlation* of the high-level VGG features, and significantly improves the realism and sharpness of frames (see Figure 4b).

Another drawback of recent interpolation methods [18,3,23,35] is training complexity, because they typically rely on scarce data to pre-train additional optical flow, depth, or other prior networks. Such data scarcity is even more critical for large motion. The DAIN approach [3], for example, incorporates a depth network, and the works in [18,23] use additional networks to estimate per-pixel motion. To simplify the training process, another contribution of this work is a unified architecture for frame interpolation, which is trainable from regular frame triplets alone.

In summary, the main contributions of our work are:

- We expand the scope of frame interpolation to a novel near-duplicate photos interpolation application, and open a new space for the community to tackle.
- We adapt a multi-scale feature extractor that shares weights, and propose a scale-agnostic bi-directional motion estimator to handle both small and large motion well, using regular training frames.
- We adopt a Gram matrix-based loss function to inpaint large disocclusions caused by large scene motion, leading to crisp and pleasing frames.
- We propose a unified, single-stage architecture, to simplify the training process and remove the reliance on additional optical flow or depth networks.

2 Related Work

Various CNN-based frame interpolation methods [11,3,20,18,6,8,35,13,22,23,32] [26,15] have been proposed to up-scale frame rate of videos. To our knowledge, no prior work exists on near-duplicate photos interpolation. We, however, summarize frame interpolation methods related to our approach.

Large Motion. Handling large motion is an important yet under-explored topic in frame interpolation. The work in [28] handles large motion by training on 4K sequences with extreme motion. While this is a viable approach, it does not generalize well on regular footage as discussed in [23]. Similarly, other approaches [18,23] perform poorly when the test motion range deviates from the training motion range. We adapt a multi-scale shared feature extractor [31,9,10], and present a “scale-agnostic” motion estimation module, which allows us to learn large and small motion with equal priority, and show favorable generalization ability in various benchmarks.

Image Quality. One of our key contributions is high quality frame synthesis, especially in large disoccluded regions caused by large motion. Prior work [20,19,21] improves image quality by learning a per-pixel kernel instead of an offset vector, which is then convolved with the inputs. While effective at improving quality, they cannot handle large motion well. Other approaches optimize models with perceptual losses [20,18]. Some consider an adversarial loss [1], albeit with a complex training process. Our work proposes to adopt the Gram matrix loss [7], which builds up on the perceptual loss and yields high quality and pleasing frames.

Single-Stage Networks. The first CNN-based supervised frame interpolators propose UNet-like networks, trained from inputs and target frames [16,11]. Recent work [3] introduces a depth network to handle occlusions, [18,23] incorporate motion estimation modules, and [30,22,35] rely on pre-trained HED [33] features. While impressive results are achieved, multiple networks can make training processes complex. It may also need scarce data to pre-train the prior networks. Pre-training datasets, e.g. optical flows, are even more scarce for large motion. Our work introduces a single unified network, trainable from regular frame triples alone, without additional priors, and achieves state-of-the-art results.

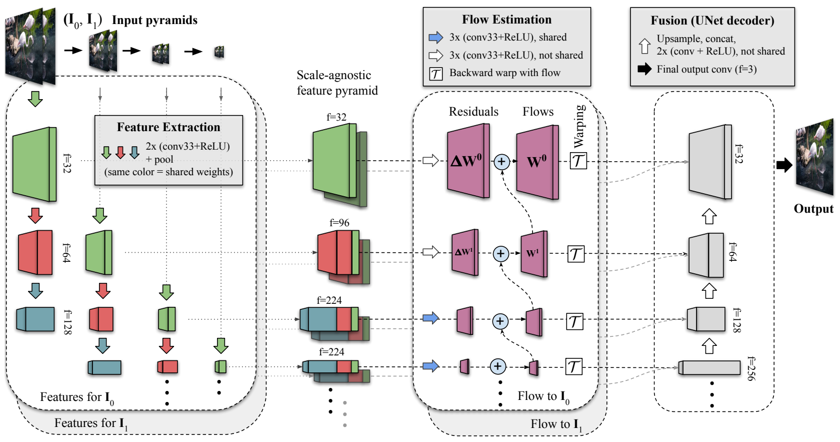


Fig. 2. FILM architecture. Our flow estimation module computes "scale agnostic" bi-directional flows based on the feature pyramids, extracted by shared weights.

3 Method

Given two input images $(\mathbf{I}_0, \mathbf{I}_1)$, with large in-between motion, we synthesize a mid-image $\hat{\mathbf{I}}_t$, with time $t \in (0, 1)$, as:

$$\hat{\mathbf{I}}_t = \mathcal{M}(\mathbf{I}_0, \mathbf{I}_1), \tag{1}$$

where $\mathcal{M}$ is our FILM network trained with a ground-truth $\mathbf{I}_t$. During training, we supervise at $t = 0.5$ and we predict more in-between images by recursively invoking FILM.

A common approach to handle large motion is to employ feature pyramids, which increases receptive fields. However, a standard pyramid learning has two difficulties: 1) small fast-moving objects disappear at coarse levels, and 2) the number of pixels is drastically smaller at coarse levels (i), $(\frac{H}{2^i} \times \frac{W}{2^i})$, which means there are fewer pixels to provide large motion supervision. To overcome these challenges, we propose to share the convolution weights across the scales. Based on the intuition that large motion at finer scales should be the same as small motion at coarser scales, sharing weights allows us to boost the number of pixels available for large motion supervision.

FILM has three main stages: Shared feature extraction, scale-agnostic motion estimation, and a fusion stage that outputs the resulting color image. Figure 2 shows an overview of FILM.

Feature Extraction. We adapt a feature extractor from [31], that allows weight sharing across the scales, to create a "scale-agnostic" feature pyramid. It is constructed in three steps as follows.

First, we create image pyramids $\{\mathbf{I}_0^l\}$ and $\{\mathbf{I}_1^l\}$ for the two input images, where $l \in [1, 7]$ is the pyramid level.

Second, starting at the image at each l-th pyramid level, we build feature pyramids (the columns in Figure 2) using a shared UNet encoder. Specifically, we extract multi-scale features $\{\mathbf{f}_0^{l,d}\}$ and $\{\mathbf{f}_1^{l,d}\}$, with $d \in [1,4]$ being the depth at that l-th level (Figure 2 only uses $d \in [1,3]$ for illustration). Mathematically,

$$\mathbf{f}_0^{l,d} = \mathcal{H}^d(\mathbf{I}_0^l), \tag{2}$$

where $\mathcal{H}^d$ is a stack of convolutions, shown in Figure 2 with the green arrow for d=1, red for d=2, and dark-blue for d=3. Note that, the same $\theta^{(\mathcal{H}^d)}$ convolution weights are shared for the same d-th depth at each pyramid level, to create *compatible multiscale features*. Each $\mathcal{H}^d$ is followed by an average pooling with a size and stride of 2.

As a third and final step of our feature extractor, we construct our scale-agnostic feature pyramids, $\{\mathbf{F}_0^l\}$ and $\{\mathbf{F}_1^l\}$, by concatenating the feature maps with different depths, but the same spatial dimensions, as:

$$\mathbf{F}_0^l = \left(\mathbf{f}_0^{l-2,d=3}, \mathbf{f}_0^{l-1,d=2}, \mathbf{f}_0^{l,d=1}\right), \tag{3}$$

and the scale-agnostic feature, $\mathbf{F}_1^l$ of $\mathbf{I}_1$, at the l-th pyramid level, can be given in a similar way by Equation 3. As shown in Figure 2, the finest level feature (green) can only aggregate one feature map, the second finest level two (green+red), and the rest can aggregate three shared feature maps.

Flow Estimation. Once we extract the feature pyramids, $\{\mathbf{F}_0^l\}$ and $\{\mathbf{F}_1^l\}$, we use them to calculate a bi-directional motion at each pyramid level. Similar to [30], we start the motion estimation from the coarsest level (in our case $l = 7$). However, in contrast to other methods, we directly predict task oriented [34,16] flows, $\mathbf{W}_{t\to 0}$ and $\mathbf{W}_{t\to 1}$, from the mid-frame to the inputs.

We compute the task oriented flow at each level $\mathbf{W}_{t\to 1}^l$ as the sum of predicted residual and the upsampled flow from the coarser level l+1, based on the intuition that large motion at finer scales should be the same as small motion at coarser scales, as:

$$\mathbf{W}_{t\to 1}^l = \left(\mathbf{W}_{t\to 1}^{l+1}\right)_{\times 2} + \mathcal{G}^l\left(\mathbf{F}_0^l, \hat{\mathbf{F}}_{t\leftarrow 1}^l\right), \tag{4}$$

where $(\bullet)_{\times 2}$ is a bilinear up-sampling, $\mathcal{G}^l$ is a stack of convolutions that estimates the residual, and $\hat{\mathbf{F}}_{t\leftarrow 1}^l$ is the backward warped scale-agnostic feature map at t=1, obtained by bilinearly warping $\mathbf{F}_1^l$ with the upsampled flow estimate, as,

$$\hat{\mathbf{F}}_{t\leftarrow 1}^l = \mathcal{T}\left(\mathbf{F}_1^l, \left(\mathbf{W}_{t\to 1}^{l+1}\right)_{\times 2}\right), \tag{5}$$

with $\mathcal{T}$ being a bilinear resample (warp) operation. Figure 2 depicts $\mathcal{G}^l$ by the blue or white arrows, depending on the pyramid level. Note that, the same residual convolution weights $\theta^{(\mathcal{G}^l)}$ are shared by levels $l \in [3,7]$.

Finally, we create the feature pyramid at the intermediate time t, $\{\mathbf{F}_{t\leftarrow 1}^l\}$ and $\{\mathbf{F}_{t\leftarrow 0}^l\}$, by backward warping the feature pyramid, at t=1 and t=0, with the flows given by Equation 4, as:

$$\mathbf{F}_{t\leftarrow 1}^l = \mathcal{T}\left(\left(\mathbf{F}_1^l, \mathbf{I}_1^l\right), \mathbf{W}_{t\to 1}^l\right), \tag{6}$$

$\mathbf{F}^l_{t\leftarrow 0}$ can be given in a similar way as Equation 6.

Fusion. The final stage of FILM concatenates, at each l-th pyramid, the scale-agnostic feature maps at t and the bi-directional motions to t, which are then fed to a UNet-like [27] decoder to synthesize the final mid-frame $\hat{\mathbf{I}}_t$. Mathematically, the fused input at each l-th decoder level is given by,

$$\left(\mathbf{F}^l_{t\leftarrow 1}, \mathbf{F}^l_{t\leftarrow 0}, \mathbf{W}^l_{t\rightarrow 0}, \mathbf{W}^l_{t\rightarrow 1}\right). \tag{7}$$

Figure 2 illustrates the decoder's convolutions and resulting activations with a white arrow and gray boxes, respectively.

3.1 Loss Functions

We use only image synthesis losses to supervise the final output of our FILM network; we do not use auxiliary losses tapped into any intermediate stages. Our image synthesis loss is a combination of three terms.

First, we use the L1 reconstruction loss that minimizes the pixel-wise RGB difference between the interpolated frame $\hat{\mathbf{I}}_t$ and the ground-truth frame $\mathbf{I}_t$, given by:

$$\mathcal{L}_1 = \|\hat{\mathbf{I}}_t - \mathbf{I}_t\|_1. \tag{8}$$

The $\mathcal{L}_1$ loss captures the motion between the inputs $(\mathbf{I}_0, \mathbf{I}_1)$ and yields interpolation results that score well on benchmarks, as is discussed in Section 5.2. However, the interpolated frames are often blurry.

Second, to enhance image details, we add a perceptual loss, using the L1 norm of the VGG-19 features [29]. The perceptual loss, also called VGG-loss, $\mathcal{L}_{\text{VGG}}$, is given by,

$$\mathcal{L}_{\text{VGG}} = \frac{1}{L}\sum_{l=1}^{L} \alpha_l \left\| \Psi_l(\hat{\mathbf{I}}_t) - \Psi_l(\mathbf{I}_t) \right\|_1, \tag{9}$$

where $\Psi_l(\mathbf{I}_i) \in \mathbb{R}^{\text{H}\times\text{W}\times\text{C}}$ is the features from the l-th selected layer of a pre-trained Imagenet VGG-19 network for $\mathbf{I}_i \in \mathbb{R}^{\text{H}\times\text{W}\times 3}$, L is the number of the finer layers considered, and α_l is an importance weight of the l-th layer.

Finally, we employ the Style loss [7,25,14] to further expand on the benefits of $\mathcal{L}_{\text{VGG}}$. The style loss $\mathcal{L}_{\text{Gram}}$, also called Gram matrix loss, is the L2 norm of the auto-correlation of the VGG-19 features [29]:

$$\mathcal{L}_{\text{Gram}} = \frac{1}{L}\sum_{l=1}^{L} \alpha_l \left\| \text{M}_l(\hat{\mathbf{I}}_t) - \text{M}_l(\mathbf{I}_t) \right\|_2, \tag{10}$$

where the Gram matrix of the interpolated frame at the l-th layer, $\text{M}_l(\hat{\mathbf{I}}_t) \in \mathbb{R}^{\text{C}\times\text{C}}$, is given by:

$$\text{M}_l(\hat{\mathbf{I}}_t) = \left(\Psi_l(\hat{\mathbf{I}}_t)\right)^{\intercal}\left(\Psi_l(\hat{\mathbf{I}}_t)\right), \tag{11}$$

and the Gram matrix of the ground-truth image, $\text{M}_l(\mathbf{I}_t)$, can be given in a similar way as Equation 11.

To our knowledge, this is the first work that applies the Gram matrix loss to frame interpolation. We found this loss to be effective in synthesizing sharp images with rich details when inpainting large disocclusion regions caused by large scene motion.

To achieve high benchmark scores as well as high quality frame synthesis, we train our models with an optimally weighted combination of the RGB, VGG and Gram matrix losses. The combined loss, which we denote $\mathcal{L}_S$, is defined as,

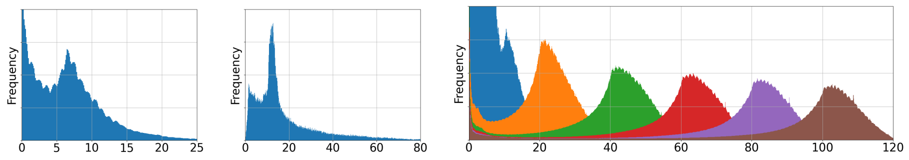
$$\mathcal{L}_S = w_l \mathcal{L}_1 + w_{\text{VGG}} \mathcal{L}_{\text{VGG}} + w_{\text{Gram}} \mathcal{L}_{\text{Gram}}, \tag{12}$$

with the weights $(w_l, w_{\text{VGG}}, w_{\text{Gram}})$ determined empirically, as detailed in the Supplementary Materials.

3.2 Large Motion Datasets

To study FILM's ability to handle large motion, we created a "bracketed" dataset containing five training sub-sets. Each containing examples with motion disparity in the following ranges, in pixels: 0-40, 0-60, 0-80, 0-100, and 0-120.

We procedurally mine 512×512 image triplets from publicly available videos, extending the method described in [5]. We apply this procedure to generate several motion brackets, i.e.: 0-20, 20-40, ..., 100-120. The motion distribution histograms of these brackets are shown overlapped in Figure 3. The effect of training using blends with increasing motion range is analyzed in Section 5.3.



(a) Vimeo motion magnitude. **(b)** Xiph motion magnitude. **(c)** Bracketed motion magnitude.

Fig. 3. Motion magnitude histograms of datasets Vimeo-90K (3a), Xiph-4K (3b) and Bracketed (3c).

4 Implementation Details

We implemented our model in TensorFlow 2. As training data, we use either Vimeo-90K or one of our large motion datasets described in Section 3.2.

For the Vimeo-90K dataset, we use a batch size of 8, with a 256×256 random crop size, distributed over 8 NVIDIA V100 GPUs. We apply data augmentation: Random rotation with [-45°,45°], rotation by multiples of 90°, horizontal flip, and reversing triplets. We use Adam [12] optimizer with β_1=0.9 and β_2=0.999, without weight decay. We use an initial learning rate of $1e^{-4}$ scheduled (piece-wise linear) with exponential decay rate of 0.464, and decay steps of 750K, for 3M iterations.

For comparison with the recent state-of-the-art models, we trained two versions: One optimized using $\mathcal{L}_1$ loss alone, which achieves higher benchmark scores, and another, that favours image quality, trained with our proposed style loss, $\mathcal{L}_S$. Our style loss optimally combines $\mathcal{L}_1$, $\mathcal{L}_{\text{VGG}}$, and $\mathcal{L}_{\text{Gram}}$.

To perform our qualitative evaluations, we also implement the SoftSplat [18] in TensorFlow 2, since pre-trained models were not available at the time of writing. In the Supplementary Materials, we show our faithful implementations on a DAVIS [24] image sample rendered in [18]. We found that renderings with our implementation to be quite comparable to the ones provided in the original paper. We provide additional implementation details in the Supplementary.

5 Results

Using existing benchmarks, we quantitatively compare FILM to recent methods: DAIN [3], AdaCoF [13], BMBC [22], SoftSplat [18], and ABME [23]. We additionally provide qualitative comparisons (to SoftSplat and ABME) on near-duplicate interpolation, for which no benchmarks currently exist.

Metrics. We use common quantitative metrics: Peak Signal-To-Noise Ratio (PSNR) and Structural Similarity Image Metric (SSIM). High PSNR and SSIM scores indicate better quality.

Datasets. We report metrics on Vimeo-90K [34], UCF101[16], Middlebury [2], and on a 4K large motion dataset Xiph [17,18]. Figure 3 shows motion magnitude histograms for Vimeo-90K and Xiph. Vimeo-90K (3a) motion is limited to 25 pixels, while the Xiph (3b) has a long-tailed distribution extending to 80 pixels.

In this comparison, all methods are with the Vimeo-90K dataset. To evaluate visual quality, we use a new challenging near-duplicate photos as the testing dataset. For ablation studies on large motion, we use our "bracketed" dataset (see Section 3.2) as the training dataset.

5.1 Quantitative Comparisons

Small-to-Medium Motion. Table 1 shows midpoint frame interpolation comparisons with DAIN [3], AdaCoF [13], BMBC [22], SoftSplat [18], and ABME [23] on small-to-medium motion datasets: Vimeo-90K, Middlebury, and UCF101.

The SoftSplat method reports two sets of results, one set trained with color loss ($\mathcal{L}_{Lap}$), which performs better on standard benchmarks, and another trained with a perceptually-sensitive loss ($\mathcal{L}_F$), which leads to perceptually higher quality frames. The rest report results obtained by training with various color or low-level loss functions.

Based on color losses, ABME outperforms all other methods on Vimeo-90K. On Middlebury and UCF101, SoftSplat trained with color loss has the highest

	Vimeo-90K [34]		**Middlebury** [2]		**UCF101** [16]	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
DAIN	34.70	0.964	36.70	0.965	35.00	0.950
AdaCoF	34.35	0.973	35.72	**0.978**	34.90	0.968
BMBC	35.01	0.976	n/a	n/a	35.15	0.969
SoftSplat-$\mathcal{L}_{Lap}$	36.10	0.970	**38.42**	0.971	**35.39**	0.952
ABME	**36.18**	**0.981**	n/a	n/a	35.38	**0.970**
Our FILM-$\mathcal{L}_1$	36.06	0.970	37.52	0.966	35.32	0.952
SoftSplat-$\mathcal{L}_F$	35.48	0.964	37.55	0.965	35.10	0.948
Our FILM-$\mathcal{L}_{\text{VGG}}$	35.76	0.967	37.43	**0.966**	**35.20**	**0.950**
Our FILM-$\mathcal{L}_S$	**35.87**	**0.968**	**37.57**	**0.966**	35.16	0.949

Table 1. Comparison on *small-to-medium motion* benchmarks. Best scores for color losses are in **blue**, for perceptually-sensitive losses in **red**. In this comparison, all methods are trained on Vimeo-90K.

PSNR. We note that ABME and SoftSplat are complex to train, each consisting of multiple sub-networks dedicated to motion estimation, refinement, or synthesis. Their training processes involve multiple datasets and stage-wise pre-training. Data scarcity, which is even more critical in large motion, could also complicate pre-training. Our unified, single-stage, FILM network achieves competitive PSNR scores.

The perception-distortion tradeoff [4] proved that minimizing distortion metrics alone, like PSNR or SSIM, can have a negative effect on the perceptual quality. As such, we also optimize our model with our proposed Gram Matrix-based loss, $\mathcal{L}_S$, which optimally favours both color differences and perceptual quality.

When including perceptually-sensitive losses, FILM outperforms the state-of-the-art SoftSplat on Vimeo-90K. We also achieve the highest scores on Middlebury and UCF101. In the next Subsection 5.2, we show visual comparisons that support the quantitative gains in PSNR with gains in image quality.

Large Motion. Table 2 presents midpoint frame interpolation comparisons on Xiph-2K and Xiph-4K, all methods (including FILM) trained on Vimeo-90K. FILM outperforms all other models for color-based losses. Note that (not shown in the table) when training FILM on a custom large motion dataset, detailed in Section 3.2, we can achieve an additional performance gain of +0.5dB on Xiph-4K, the benchmark with the largest motions.

When including perceptually-sensitive losses, FILM outperforms SoftSplat-$\mathcal{L}_F$ in PSNR on Xiph-2K and both PSNR and SSIM on the larger motion Xiph-4K. Thus, FILM is better able to generalize from the small motions in the Vimeo-90K training datasets to the larger motions present in the Xiph test sets. We hope these findings will interest the greater research community, where large motion is often challenging. In the next Subsection 5.2, we provide visual results that support the effectiveness of our method in samples with motion ranges as large as 100 pixels.

	Xiph-2K [17]		**Xiph-4K** [17]	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑
DAIN	35.95	0.940	33.49	0.895
ToFlow	33.93	0.922	30.74	0.856
AdaCoF	34.86	0.928	31.68	0.870
BMBC	32.82	0.928	31.19	0.880
ABME	36.53	0.944	33.73	0.901
SoftSplat-$\mathcal{L}_{Lap}$	36.62	0.944	33.60	0.901
Our FILM-$\mathcal{L}_1$	**36.66**	**0.951**	**33.78**	**0.906**
SoftSplat-$\mathcal{L}_F$	35.74	**0.944**	32.55	0.865
Our FILM-$\mathcal{L}_S$	**36.38**	0.942	**33.29**	**0.882**

Table 2. Comparison on *large motion* benchmarks. Best scores for color losses in (**blue**), and for perceptually-sensitive losses in (**red**). In this comparison, all methods are trained on Vimeo-90K

.

5.2 Qualitative Comparisons

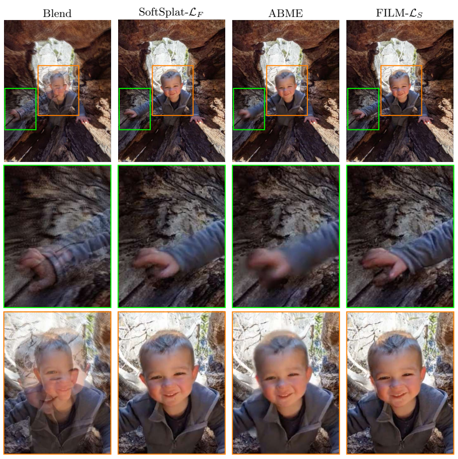
We provide visual results that support our quantitative results. We use the version of the model that yields high image quality, i.e.: our FILM-$\mathcal{L}_S$ and SoftSplat-$\mathcal{L}_F$. For ABME [3], we create visual results using the released pre-trained models. For SoftSplat[4] [18], we use our faithful implementation, since neither source code nor pre-trained model was publicly available at the time of writing.

Sharpness. To evaluate the effectiveness of our Gram Matrix-based loss function (Equation 11) in preserving image sharpness, we visually compare our results against images rendered with other methods. As seen in Figure 4a, our method synthesizes visually superior results, with crisp image details on the face and preserving the articulating fingers.

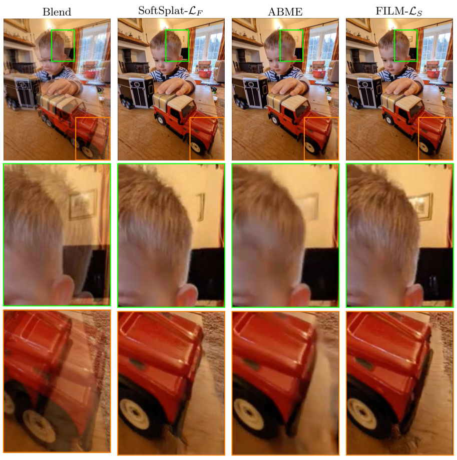
Disocclusion Inpainting. To effectively inpaint disoccluded pixels, models must learn appropriate motions or hallucinate novel pixels, this is especially critical in large scene motion, which causes wide disocclusions. Figure 4b shows different methods, including ours, inpainting large disocclusions. Compared to the other approaches, FILM correctly paints the pixels while maintaining high fidelity. It also preserves the structure of objects, e.g. the red toy car, while SoftSplat [18] shows deformation, and ABME [23] creates blurry inpainting.

Large Motion. Large motion is one of the most challenging aspects of frame interpolation. Figure 5 shows results for different methods on a sample with 100 pixels disparity. Both SoftSplat [18] and ABME [23] were able to capture motions near the dog's nose, however they create large artifacts on the ground. FILM's strength is seen capturing the motion well and keeping the background details. Please see our Supplementary Materials for more visual results.

[3] https://github.com/JunHeum/ABME
[4] https://github.com/sniklaus/softmax-splatting



(a) Sharpness.



(b) Inpainting disocclusions.

Fig. 4. Qualitative comparison on sharpness and large disocclusion inpainting. (4a) FILM produces sharp images and preserves the fingers. SoftSplat [18] shows artifacts (fingers) and ABME [23] has blurriness (the face). (4b) Our method inpaints large disocclusions well, because of our proposed Gram Matrix-based (Style) loss. SoftSplat [18] and ABME [23] produce blurry inpaintings or unnatural deformations.

5.3 Ablations

In this section, we present ablation studies to analyze the design choices of FILM.

Weight Sharing. We compare our shared feature extractor with a regular UNet-like encoder [27] that uses independent weights at all scales, forcing us to also learn independent flows. Table 3 presents mid-frame results in PSNR. It is not straightforward to construct models that are fair to compare: one could either match the total number of weights or the number of filters at each level. We chose to use a UNet encoder that starts from the same number of filters as ours and then doubles the number at each level. The FILM-model we have used in this paper starts with 64 filters, so this leads to a UNet with feature counts: $[64, 128, 256...]$. We find that training with this configuration does not converge without weight sharing. To study this further, we construct two simpler variants of our model, starting from 32 filters. We are able to train these two models with a small loss in PSNR as compared to the equivalent model that shares weights.

To conclude, weight sharing allows training a more powerful model, reaching a higher PSNR. Additionally, sharing may be important for fitting models in GPU memory in practical applications. Further, the model with weight sharing is visually superior with substantially better generalization when testing on pairs with motion magnitude beyond the range in the training data (see Figure 6).

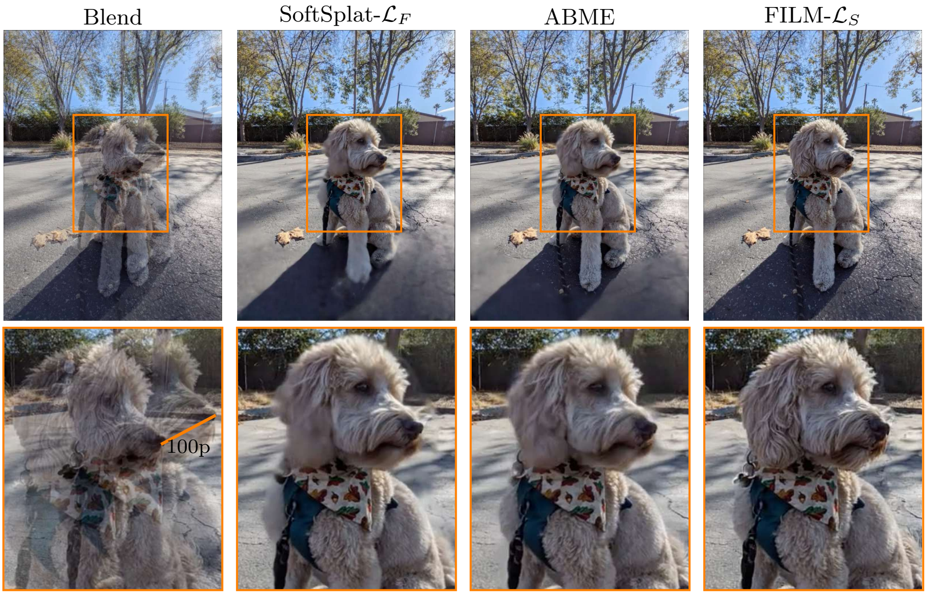


Fig. 5. Qualitative comparison on large motion. Inputs with 100pixels disparity overlaid (left). Although both SoftSplat [18], ABME [23] capture the motion on the dog's nose, they appear blurry, and create a large artifact on the ground. FILM's strength is seen capturing the motion well and maintaining the background details.

Gram Matrix (Style) Loss. Figure 7 presents qualitative results of FILM trained with $\mathcal{L}_1$, adding $\mathcal{L}_{VGG}$, and with our proposed style loss $\mathcal{L}_S$, given by Equation 12. Using $\mathcal{L}_1$ alone leads to blurry images (red box), while adding $\mathcal{L}_{VGG}$ loss reduces blurry artifacts (orange box). Our proposed loss (green box) significantly improves sharpness of our FILM method.

Motion Ranges. We study the effect of the training dataset motion range on the model's ability to handle different motions at test time, using the "bracketed dataset" (see Section 3.2). For this study, we use a reduced FILM-med to save compute resources. FILM-med is trained on motion ranges of 0-40, 0-60, 0-80, 0-100 and 0-120 pixels. They are evaluated on Vimeo-90K (0-25 range) and Xiph-

model	**PSNR (w/ sharing)**	**PSNR (w/o sharing)**
FILM	36.06	N/A
FILM-med	35.30	35.28
FILM-lite	35.09	34.93

Table 3. Weight sharing ablation study. A model without multi-scale feature sharing achieves results that are slightly lower than those achieved with shared features, e.g. FILM-med and FILM-lite. We have not been able to train our highest quality model (FILM) without weight sharing as the training gets unstable (indicated with N/A).

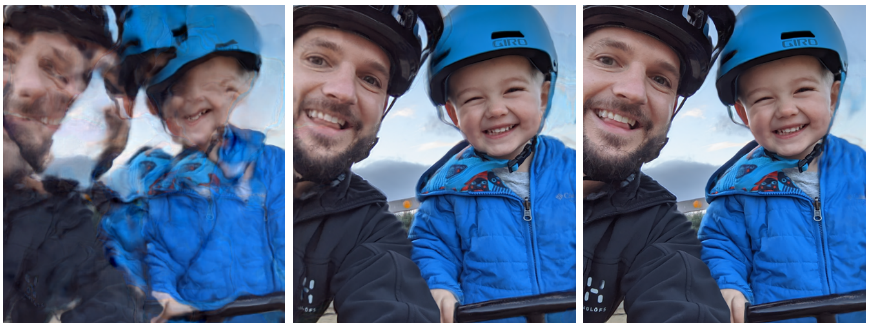


Fig. 6. The visual impact of sharing weights. **Left**: no sharing (FILM-med), **Middle**: sharing (FILM-med), **Right**: sharing (FILM), i.e., the highest quality model. Sharing weights is clearly better. Quality and sharpness increases when going from the medium to the full model with sharing.

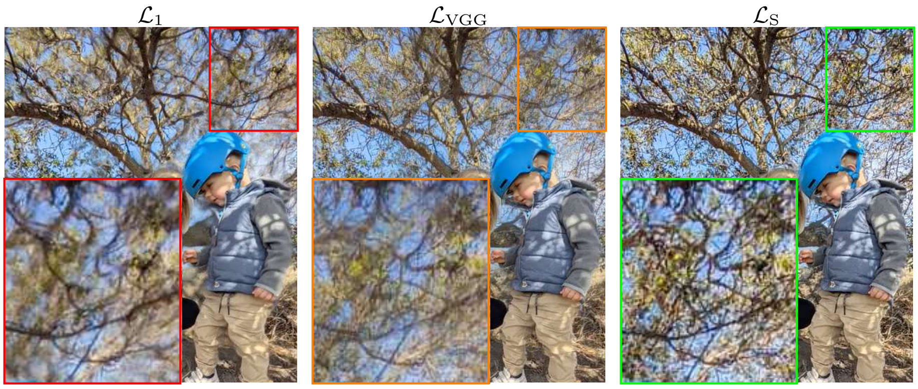


Fig. 7. Loss function comparison on our FILM. L1 loss (left), L1 plus VGG loss (middle), and our proposed style loss (right), showing significant sharpness improvements (green box).

4K (0-80 range), as shown in Figure 8. On Vimeo-90K, FILM-med trained with the smallest motion range performs the best. As larger motion is added, PSNR goes down significantly.

We hypothesize that this behavior is caused by two factors: 1) when training for larger motion, the model assumes a larger search window for correspondence, and thus has more chances to make errors. 2) with larger motion, the problem is simply harder and less neural capacity is left for smaller motion.

On Xiph-4K, the best performance is obtained by training with motion range 0-100 pixels. Motivated by our finding, we trained our best model (FILM) with this dataset, and achieve an additional PSNR performance gain of +0.5dB over the state-of-the-art, as described in Section 5.1.

In summary, our findings indicate that scale-agnostic features and shared weight flow prediction improve the model's ability to learn and generalize to a wider range of motion. In addition, we find that best results are obtained when the training data also matches the test-time motion distribution.

5.4 Performance and Memory

Table 4 presents inference times and memory comparisons on an NVIDIA V100 GPU. We report the average of 100 runs. Our FILM is 3.95× faster than ABME, and 9.75× faster than SoftSplat, while using only 1.27× and 1.01× more memory than ABME and SoftSplat, respectively. Our model is slightly larger due to its deep pyramid (7 levels), but the time performance gains are significantly large.

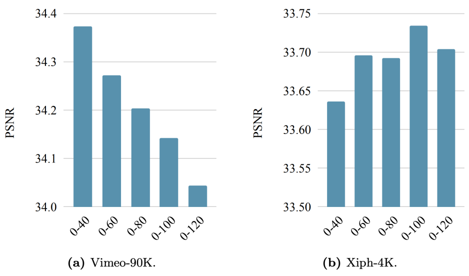


Fig. 8. A study of the effect of the training dataset's motion range on the PSNR, when evaluated on (a) Vimeo-90K, and (b) Xiph-4K.

5.5 Limitations

In some instances, FILM produces un-natural deformations when the in-between motion is extreme. Although resulting videos are appealing, the subtle movements may not look natural. We provide failure examples in our Supplementary video.

Interpolation Method	Inference Time (Second)↓	Peak Memory (GB)↓
SoftSplat	3.834	4.405
ABME	1.554	**3.506**
Our FILM	**0.393**	4.484

Table 4. Inference time and memory comparison for a 720p frame interpolation.

6 Conclusions

We have introduced an algorithm for large motion frame interpolation (FILM), in particular, for near-duplicate photos interpolation. FILM is a simple, unified and single-stage model, trainable from regular frames, and does not require additional optical-flow or depth prior networks, or their scarce pre-training data. Its core components are a feature pyramid that shares weight across scales and a "scale-agnostic" bi-directional motion estimator that learns from frames with normal motion but generalizes well to frames with large motion. To handle wide disocclusions caused by large motion, we optimize our models with the Gram matrix loss that matches the correlation of features to generate sharp frames. Extensive experimental results show that FILM outperforms other methods on large motions while still handling small motions well, and generates high quality and temporally smooth videos. Source codes and pre-trained models are available at https://film-net.github.io.

Acknowledgements. We thank Orly Liba and Charles Herrmann for feedback on the text, Jamie Aspinall for the imagery in the paper, Dominik Kaeser, Yael Pritch, Michael Nechyba and David Salesin for support.

References

1. van Amersfoort, J., Shi, W., Acosta, A., Massa, F., Totz, J., Wang, Z., Caballero, J.: Frame interpolation with multi-scale deep loss functions and generative adversarial networks. arXiv preprint arXiv:1711.06045 (2017) 3
2. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. International journal of computer vision **92**(1), 1–31 (2011) 2, 8, 9
3. Bao, W., Lai, W.S., Ma, C., Zhang, X., Gao, Z., Yang, M.H.: Depth-aware video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3703–3712 (2019) 2, 3, 8
4. Blau, Y., Michaeli, T.: The perception-distortion tradeoff. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6228–6237 (2018) 9
5. Brooks, T., Barron, J.T.: Learning to synthesize motion blur. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6840–6848 (2019) 7
6. Ding, T., Liang, L., Zhu, Z., Zharkov, I.: Cdfi: Compression-driven network design for frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8001–8011 (2021) 3
7. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2414–2423 (2016) 3, 6
8. Huang, Z., Zhang, T., Heng, W., Shi, B., Zhou, S.: Rife: Real-time intermediate flow estimation for video frame interpolation. arXiv preprint arXiv:2011.06294 (2020) 2, 3
9. Hur, J., Roth, S.: Iterative residual refinement for joint optical flow and occlusion estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5754–5763 (2019) 3
10. Jiang, H., Sun, D., Jampani, V., Lv, Z., Learned-Miller, E., Kautz, J.: Sense: A shared encoder network for scene-flow estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3195–3204 (2019) 3
11. Jiang, H., Sun, D., Jampani, V., Yang, M.H., Learned-Miller, E., Kautz, J.: Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9000–9008 (2018) 3
12. KingaD, A.: A methodforstochasticoptimization. Anon. InternationalConferenceon Learning Representations. SanDego: ICLR (2015) 7
13. Lee, H., Kim, T., Chung, T.y., Pak, D., Ban, Y., Lee, S.: Adacof: Adaptive collaboration of flows for video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5316–5325 (2020) 3, 8
14. Liu, G., Reda, F.A., Shih, K.J., Wang, T.C., Tao, A., Catanzaro, B.: Image inpainting for irregular holes using partial convolutions. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 85–100 (2018) 6

15. Liu, Y.L., Liao, Y.T., Lin, Y.Y., Chuang, Y.Y.: Deep video frame interpolation using cyclic frame generation. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 8794–8802 (2019) 3
16. Liu, Z., Yeh, R.A., Tang, X., Liu, Y., Agarwala, A.: Video frame synthesis using deep voxel flow. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4463–4471 (2017) 2, 3, 5, 8, 9
17. Montgomery, C., et al.: Xiph. org video test media (derf's collection), the xiph open source community, 1994. Online, https://media. xiph. org/video/derf **3** (1994) 8, 10
18. Niklaus, S., Liu, F.: Softmax splatting for video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5437–5446 (2020) 2, 3, 8, 10, 11, 12
19. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive convolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 670–679 (2017) 3
20. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive separable convolution. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 261–270 (2017) 3
21. Niklaus, S., Mai, L., Wang, O.: Revisiting adaptive convolutions for video frame interpolation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1099–1109 (2021) 3
22. Park, J., Ko, K., Lee, C., Kim, C.S.: Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16. pp. 109–125. Springer (2020) 3, 8
23. Park, J., Lee, C., Kim, C.S.: Asymmetric bilateral motion estimation for video frame interpolation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14539–14548 (2021) 2, 3, 8, 10, 11, 12
24. Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., Sorkine-Hornung, A.: A benchmark dataset and evaluation methodology for video object segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 724–732 (2016) 8
25. Reda, F.A., Liu, G., Shih, K.J., Kirby, R., Barker, J., Tarjan, D., Tao, A., Catanzaro, B.: Sdc-net: Video prediction using spatially-displaced convolution. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 718–733 (2018) 6
26. Reda, F.A., Sun, D., Dundar, A., Shoeybi, M., Liu, G., Shih, K.J., Tao, A., Kautz, J., Catanzaro, B.: Unsupervised video interpolation using cycle consistency. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 892–900 (2019) 3
27. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015) 6, 11
28. Sim, H., Oh, J., Kim, M.: Xvfi: Extreme video frame interpolation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14489–14498 (2021) 2, 3
29. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) 6
30. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In: CVPR (June 2018) 3, 5

31. Trinidad, M.C., Brualla, R.M., Kainz, F., Kontkanen, J.: Multi-view image fusion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4101–4110 (2019) 2, 3, 4
32. Xiang, X., Tian, Y., Zhang, Y., Fu, Y., Allebach, J.P., Xu, C.: Zooming slow-mo: Fast and accurate one-stage space-time video super-resolution. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 3370–3379 (2020) 3
33. "Xie, S., Tu, Z.: Holistically-nested edge detection. In: Proceedings of IEEE International Conference on Computer Vision (2015) 3
34. Xue, T., Chen, B., Wu, J., Wei, D., Freeman, W.T.: Video enhancement with task-oriented flow. International Journal of Computer Vision **127**(8), 1106–1125 (2019) 2, 5, 8, 9
35. Zhang, H., Zhao, Y., Wang, R.: A flexible recurrent residual pyramid network for video frame interpolation. In: European Conference on Computer Vision. pp. 474–491. Springer (2020) 2, 3

Supplementary Materials
FILM: Frame Interpolation for Large Motion

Fitsum Reda[1], Janne Kontkanen[1], Eric Tabellion[1], Deqing Sun[1],
Caroline Pantofaru[1], and Brian Curless[1,2]

[1] Google Research
[2] University of Washington

We provide additional implementation details and a supplementary video (visit https://film-net.github.io) for a quick overview of our paper, the motivations, as well as more visual results.

A Implementation Details

A.1 Loss Combination Weights

Our style loss ($\mathcal{L}_{\text{S}}$) optimally combines $\mathcal{L}_1$, $\mathcal{L}_{\text{VGG}}$, and $\mathcal{L}_{\text{Gram}}$. We use a piecewise linear weight-schedule to select a weight for each loss at each iteration. Specifically, we assign weights of $(1.0, 1.0, 0.0)$ for 1.5M iterations, and weights of $(1.0, 0.25, 40.0)$ for the last 1.5M iterations. For the last 1.5M iterations, the loss weights are empirically selected such that each loss contributes equally to the combined Style loss.

A.2 SoftSplat Implementation

As described in the paper, we have implemented the SoftSplat[18] in Tensorflow 2, using the author's tuned hyper-parameters, and we have been able to replicate their published benchmark scores. Note that, SoftSplat[3] provides only a CuPy implementation of the softmax splatting operator.

In Figure 1, we show our faithful implementations on a DAVIS [34] image sample rendered in [18]. We found that renderings with our implementation to be quite comparable to the ones provided in the original paper.

B Supplementary Video

Please watch the enclosed video or visit https://film-net.github.io. We have included motivations, illustrations of our methods, and more visual results and failure samples.

[3] https://github.com/sniklaus/softmax-splatting

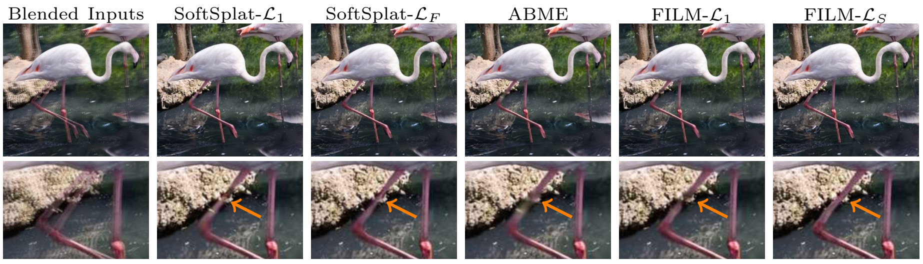


Fig. 1. Frame interpolation example from DAVIS-dataset [24] featuring a walking flamingo. From left to right: The input frames overlaid, SoftSplat-$\mathcal{L}_1$ [18], SoftSplat-$\mathcal{L}_F$ [18], ABME [23], our FILM-$\mathcal{L}_1$, and our FILM-$\mathcal{L}_S$. FILM-$\mathcal{L}_S$ produces crisp frame, while color distortions and transparencies can be seen in ABME and SoftSplat, respectively. SoftSplat renderings are generated from our faithful implementations, which we found to be quite comparable to the original paper [18].