Many-to-many Splatting for Efficient Video Frame Interpolation

Ping Hu[1*] Simon Niklaus[2] Stan Sclaroff[1] Kate Saenko[1,3]
[1]Boston University [2]Adobe Research [3]MIT-IBM Watson AI Lab

Abstract

Motion-based video frame interpolation commonly relies on optical flow to warp pixels from the inputs to the desired interpolation instant. Yet due to the inherent challenges of motion estimation (e.g. occlusions and discontinuities), most state-of-the-art interpolation approaches require subsequent refinement of the warped result to generate satisfying outputs, which drastically decreases the efficiency for multi-frame interpolation. In this work, we propose a fully differentiable Many-to-Many (M2M) splatting framework to interpolate frames efficiently. Specifically, given a frame pair, we estimate multiple bidirectional flows to directly forward warp the pixels to the desired time step, and then fuse any overlapping pixels. In doing so, each source pixel renders multiple target pixels and each target pixel can be synthesized from a larger area of visual context. This establishes a many-to-many splatting scheme with robustness to artifacts like holes. Moreover, for each input frame pair, M2M only performs motion estimation once and has a minuscule computational overhead when interpolating an arbitrary number of in-between frames, hence achieving fast multi-frame interpolation. We conducted extensive experiments to analyze M2M, and found that it significantly improves the efficiency while maintaining high effectiveness.

1. Introduction

Video frame interpolation (VFI) aims to increase frame rates of videos by synthesizing intermediate frames in between the original ones [1, 40]. As a classic problem in video processing, VFI contributes to many practical applications, including slow-motion animation [12], video editing [22], video compression [45], *etc.* In recent years, a plethora of techniques for video frame interpolation have been proposed [18,23,24,36,44,47,51]. However, frame interpolation remains an unsolved problem due to challenges like occlusions, large motion, and lighting changes.

The referenced research can roughly be categorized into motion-free and motion-based, depending on whether or not cues like optical flow are incorporated [15,42]. Motion-free models typically rely on kernel prediction [6, 9, 31, 34] or spatio-temporal decoding [7, 8, 13], which are effective but limited to interpolating frames at fixed time steps and their runtime increases linearly in the number of desired output frames. On the other end of the spectrum, motion-based approaches establish dense correspondences between frames and apply warping to render the intermediate pixels.

A common motion-based technique estimates bilateral flow for the desired time step and then synthesizes the intermediate frame via backward warping [2, 11, 12, 32, 33]. The estimation of bilateral motion is challenging though and incorrect flows can easily degrade the interpolation quality. As a result, for each time step, these methods typically apply a synthesis network to refine the bilateral flows. Another motion-based solution is to forward warp pixels to the desired time step via optical flow [1]. However, forward warping is subject to holes and ambiguities where multiple pixels map to the same location. Therefore, image refinement networks are commonly adopted to correct remaining artifacts [27, 28, 46]. However, both of these approaches re-

[*]Work primarily done while Ping was interning at Adobe.

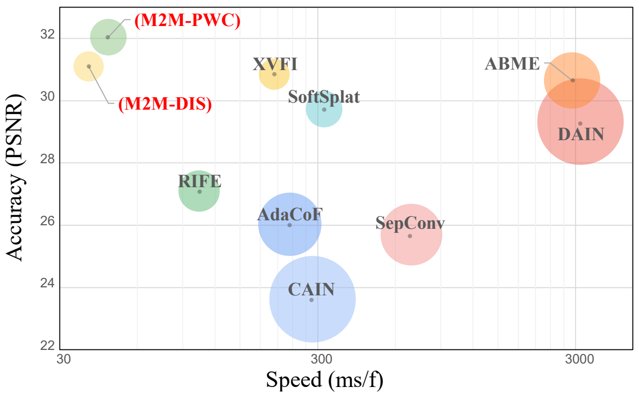


Figure 1. Performance for $\times 8$ interpolation on a "2K" version of X-TEST [39]. Runtimes for all methods were measured using a Titan X GPU. The size of each circle indicates the number of model parameters. Results for related methods include RIFE [11], SoftSplat [28], AdaCof [16], SepConv [30], XVFI [39], DAIN [2], ABME [33], and CAIN [32]. We evaluate our proposed M2M splatting using two different off-the-shelf flow estimators, "PWC" denoting PWC-Net [42] and "DIS" denoting DISFLow [15].

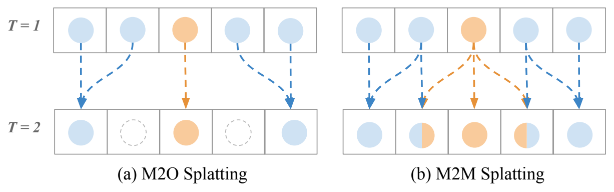


Figure 2. (a) Many-to-one splatting versus (b) many-to-many splatting for zooming motion in a scene containing blue and orange pixels. M2O splatting may results in holes, while M2M splatting allows for a more flexible image formation model.

quire significant amounts of compute, and the refinement networks need to be executed for each of the desired interpolation instants. This decreases their efficiency in multi-frame interpolation tasks since their runtime increases linearly in the number of desired output frames.

We address these challenges and strive for efficiency with a Many-to-Many (M2M) splatting framework. Specifically, our proposed M2M splatting estimates multiple bidirectional flow fields and then efficiently forward warps the input images to the desired time step before fusing any overlapping pixels. Since we directly operate on pixel colors, the quality and resolution of the underlying optical flow play a critical role. For this reason, we first apply an off-the-shelf optical flow estimator [15, 42] to extract the inter-frame motion between the two input frames at a coarse level. Based on this low-resolution optical flow estimate, a Motion Refinement Network (MRN) predicts multiple flow vectors for each pixel at the full-resolution which we then use for our image synthesis through many-splatting.

Conventional motion-based frame interpolation methods only estimate one inter-frame motion vector for each pixel [2, 11, 27, 28, 32, 33, 46]. However and as shown in Fig. 2 (a), forward warping with such a motion field manifests as many-to-one splatting, leaving unnecessary holes in the warped result. To overcome this limitation, we model a many-to-many relationship among pixels by predicting multiple motion vectors for each of the input pixels, and then forward warping the pixels to multiple locations at the desired time step. As shown in Fig. 2 (b), many-to-many splatting allows for more complex interactions among pixels, *i.e.* each source pixel is allowed to render multiple target pixels and each target pixel can be synthesized with a larger area of visual context. Unsurprisingly, many-to-many splatting leads to many more overlapping pixels. To merge these, we further introduce a learning-based fusion strategy which adaptively combines pixels that map to the same location.

Since the optical flow estimation step in our pipeline predicts time-invariant correspondence estimates, it only needs to be performed once for a given input frame pair. Once the many-to-many inter-frame motion has been established, generating new in-between frames only requires warping and fusing the input images. This is in stark contrast to previous approaches that leverage refinement networks [27, 28], allowing us to perform multi-frame interpolation an order of magnitude faster as shown in Fig. 1.

In summary, we propose 1) a Motion-Refinement Network that estimates a many-to-many relationship between the two input images, 2) a learning-based pixel fusion strategy which resolves ambiguities between overlapping pixels, and 3) a well-motivated Many-to-Many (M2M) splatting synthesis model for efficient and effective frame interpolation. Our experiments demonstrate that M2M achieves high effectiveness with fast speed, e.g. $\sim$40 ms/f using a Titan X to perform $\times 8$ interpolation of 2K videos.

2. Related Work

Motion-based video frame interpolation approaches typically estimate optical flows [15, 42] from given frames, and then propagate pixels/features to the desired target time step [26, 46, 48, 49]. Forward warping is an efficient solution to achieve this goal [1]. With bidirectional optical flow between given frames, Niklaus *et al.* [27] directly forward warp the images as well as contextual features to the interpolation instant before utilizing a synthesis network to render the output frame. To make this splatting fully differentiable, they further introduce softmax splatting [28] which allows them to train the feature extraction end-to-end. Splatting has its downsides though, since it is not only necessary to address ambiguities of multiple pixels mapping to the same location but it is also necessary to handle the holes that are present in the sparse result.

To avoid having to handle these challenges, some methods are based on backward warping instead [3, 39]. The necessary bilateral flow can, for example, be approximated from off-the-shelf flow estimates through a neural network [12] or depth-based splatting [2]. Park *et al.* [32, 33] extend these ideas and introduce a network to further improve the motion representations while Huang *et al.* [11] learn to directly estimate bilateral flows. However, estimating bilateral flow is still challenging and the backward warped pixels may still suffer from artifacts. As a result, these methods also rely on image synthesis networks to improve the interpolation quality [11, 27, 28, 32, 33]. Though shown to be effective, the bilateral flow estimation and the image synthesis networks need to be fully executed for each desired output, leading to a linearly increasing runtime when interpolating more than one in-between frame.

In contrast to these methods, our M2M approach relies on many-to-many splatting to address the issues with forward warping without relying on an image synthesis network or bilateral flow approximation/estimation.

Another dominant research direction for VFI aims to avoid explicit motion estimation altogether. One popular approach is to resample input pixels with spatially adaptive filters [19, 34]. Niklaus *et al.* [29] estimate spatially-varying

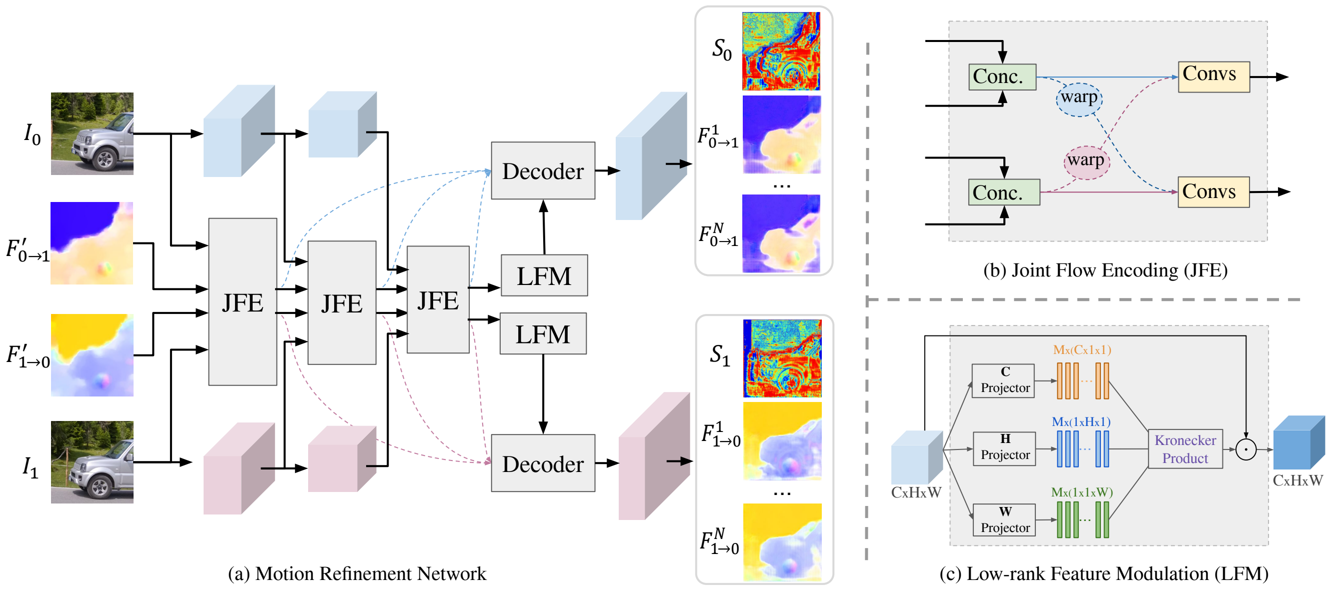


Figure 3. Overview of the (a) Motion Refinement Network and its core modules: (b) Joint Flow Encoding and (c) Low-rank Feature Modulation. Given an image pair $\{I_0, I_1\}$ and the initial bidirectional inter-frame flow $\{F'_{0\to1}, F'_{1\to0}\}$, the goal is to generate multiple refined bidirectional flows $\{F^i_{0\to1}, F^i_{1\to0}\}^N_{i=1}$ and the color reliability maps $\{S_0, S_1\}$. The "warp" in the JFE denotes backward warping.

kernels which in subsequent work are decomposed into separable kernels [30, 31], which also formulate a many-to-many correlations between pixels. However, as local patches suffer from a limited spatial range, deformable convolutions are introduced to handle large motion [6, 16]. To improve model efficiency, Ding *et al.* [9] introduce model compression [16]. Spatio-temporal decoding methods are also proposed to directly convert spatio-temporal features into target frames via channel attention [7, 8] or 3D convolutions [13]. However, most of these methods generate outputs at a fixed time, typically halfway between the input images, which limits arbitrary-time interpolation and linearly increases the runtime for multi-frame interpolation.

3. Many-to-many Splatting Framework

In this section, we describe our Many-to-Many (M2M) splatting framework for video frame interpolation. Given an input frame pair, we first estimate the bidirectional motion with an off-the-shelf method [15,42]. A Motion Refinement Network (Fig. 3 (a)) then takes the off-the-shelf motion predictions as input and estimates multiple motion vectors as well as a reliability score for each individual pixel in the input frames. Lastly, all input pixels are forward warped to the desired target time step several times via each of the multiple motion vectors, and finally merged to generate the output via a pixel fusion that leverages the estimated reliability score. With full end-to-end supervision, our M2M framework is able to achieve not only efficiency but also effectiveness. In the following, we first present the Motion Refinement Network in Sec. 3.1, then introduce the multi-splatting and fusion of pixels in Sec. 3.2.

3.1. Motion Refinement Network

Optical flow is a common technique to model inter-frame motion in videos. Yet directly applying an off-the-shelf optical flow estimator and forward warping pixels based on this estimate may be challenging. Optical flow only models a single motion vector for each pixel, thus limiting the area that a pixel can splat to and thus potentially causing holes. Moreover, most optical flow estimators are supervised with training data at a relatively low resolution and forcing them to process high-resolution frames may yield poor results. In contrast, we present the Motion Refinement Network (MRN) to upsample and refine an off-the-shelf optical flow estimate while predicting multiple motion vectors per pixel. As shown in Fig. 3 (a), the MRN pipeline is composed of three parts: Motion Feature Encoding, Low-rank Feature Modulation, and Output Decoding.

Motion Feature Encoding aims to encode multi-stage motion features from the input frames $\{I_0, I_1\}$ as well as the optical flow $\{F'_{0\to1}, F'_{1\to0}\}$ estimated by an off-the-shelf estimator [15, 42] at a coarse resolution. As outlined in Fig. 3 (a), the encoding process is designed in a hierarchical manner. At first, we extract two L-level image feature pyramids from I_0 and I_1, with the zeroth-level being the images themselves. To generate the feature representations at each pyramid level, we utilize two convolutional layers with intermittent PReLU activations to downsample the features from the previous level by a factor of two. In our implementation, we use $L = 4$, and the numbers of feature channels from shallow to deep are 16, 32, 64, and 128 respectively.

Then, from the zeroth to the last level, we apply Joint Flow Encoding (JFE) modules as illustrated in Fig. 3 (b) to progressively generate motion feature pyramids for the bidirectional flow fields $F'_{0\to1}$ and $F'_{1\to0}$. In the l-th level's JFE module, the motion and image features from the previous level are warped towards each other. Specifically, the features from the pyramid corresponding to I_0 are warped towards I_1 and vice versa using the off-the-shelf optical flow estimates. Then, the original features and the warped features are combined and downsampled using a two-layer CNN to encode the l-th level's motion features.

Low-rank Feature Modulation is designed to further enhance the motion feature representations with a low-rank constraint. The idea behind this module is that flow fields of natural dynamic scenes are highly structured due to the underlying physical constraints, which can be exploited by low-rank models to enhance the motion estimation quality [10, 37, 38, 43]. To avoid formulating explicit optimization objectives like in previous methods, which may be inefficient in high-resolution applications, we draw inspirations from Canonical Polyadic (CP) decomposition [14] and construct an efficient low-rank modulation module to enhance each flow's feature maps with low-rank characteristics.

As shown in Fig. 3 (c), given an input feature map of size $C \times H \times W$, three groups of projectors are adopted to respectively shrink the feature maps into the *channel*, *height*, and *width* dimensions. Each projector is composed of a pooling layer, 1×1 conv layers, and a sigmoid function. We apply M projectors for each of the three dimensions which results in three groups of 1-D features, whose sizes can be represented as $M \times (C \times 1 \times 1)$ for the *channel* dimension, $M \times (1 \times H \times 1)$ for the *height* dimension, and $M \times (1 \times 1 \times W)$ for the *width* dimension. Then, for each of the M vectors from the three dimensions, we apply the *Kronecker Product* to get a rank-1 tensor, whose shape is $C \times H \times W$. The M rank-1 tensors are later averaged point-wise. To ensure low-rank characteristic, M is set to be smaller than C, H, and W (we adopt $M = 16$ in this work). We combine the input features and the low-rank tensor via point-wise multiplication, where the latter serves as weights to modulate the former with low-rank characteristics.

Deep learning-based low-rank constraints have also been utilized for model compression [35], segmentation [5] and image reconstruction [50]. In this work we explore the application to motion modeling and demonstrate its effectiveness on the task of video frame interpolation.

Output Decoding generates N motion vectors as well as the reliability scores for each input pixel based on the motion feature pyramids and the feature maps subject to the low-rank prior. We adopt deconv layers to enlarge the spatial size of the feature maps. That is, the decoder operates in L stages from coarse to fine while leveraging the features encoded by the JFE modules. At the last decoding stage, the full-resolution feature maps for the flow in each direction are converted into multiple fields $\{F^i_{0\to1}, F^i_{1\to0}\}^N_{i=1}$ as well as the corresponding reliability maps $\{S_0, S_1\}$, which are later utilized to fuse pixels that map to the same location when generating the new in-between frames. An example of these outputs is visualized in Fig. 4.

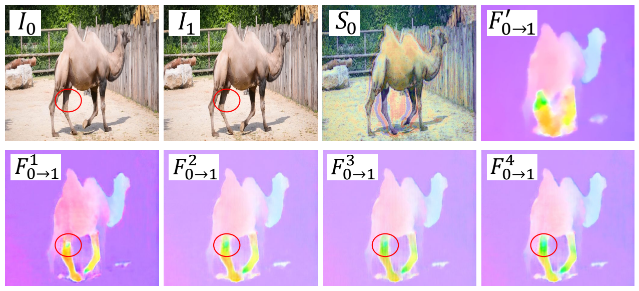


Figure 4. Examples of the MRN's output ($N = 4$). S_0 shows low reliability in areas with complex motion as intuitively expected. $\{F^N_{0\to1}\}^4_{n=1}$ refine the initial flow $F'_{0\to1}$ with better details, and decompose complex motion with shade changes (as indicated by the red circle) into multiple motion fields.

3.2. Pixel Warping and Fusion

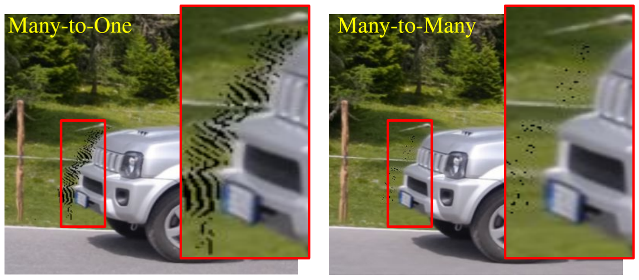
The previously estimated multi-motion fields are first used to forward warp pixels to a given target time step. Later, we present a fusion strategy to combine the colors of overlapping pixels in the output. Since both the warping and fusion steps operate with pixels' colors without any subsequent post-processing steps, an intermediate frame can be interpolated with minuscule computational overhead.

Pixel Warping. So far, we have generated N full-resolution bidirectional motion fields $\{F^n_{0\to1}, F^n_{1\to0}\}^N_{n=1}$ and pixel-wise reliability scores $\{S_0, S_1\}$ for the input video frame pair $\{I_0, I_1\}$. The next step is to synthesize an intermediate frame I_t at the desired time step $t \in (0, 1)$. Under the assumption of linear motion, we first scale each pixel's motion vectors by the desired interpolation time t as:

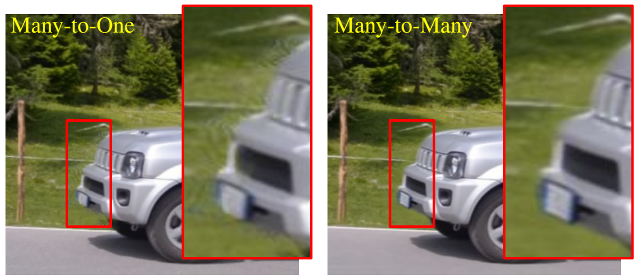
$$\begin{aligned} F^n_{0\to t}(i_0) &= t \cdot F^n_{0\to1}(i_0) \\ F^n_{1\to t}(i_1) &= (1-t) \cdot F^n_{1\to0}(i_1) \end{aligned} \tag{1}$$

where i_0 and i_1 denote the i-th source pixel in I_0 and I_1 respectively. Then, a source pixel i_s is forward warped by its n-th motion vector to $i^n_{s\to t} = \phi_F(i_s, F^n_{s\to t})$ at the desired intermediate time t, with $s \in \{0, 1\}$ representing the source frame, ϕ_F is the forward warping operation, and $F^n_{s\to t}$ is the n-th sub-motion vector of i_s as defined in Eq. 1.

We first consider utilizing a single motion vector for warping, which means each pixel is only warped to one location in the target frame. In dynamic scenes, the motion vectors may overlap with each other thus resulting in



(a) Single source frame



(b) Two source frames

Figure 5. Visualization of forward warping via many-to-one (M2O) splatting and many-to-many (M2M) splatting. (a) With one source frame, M2M splatting suffers less from banding artifacts and provides improved robustness to ambiguities near the boundaries of discontinuous motion. (b) Banding artifacts can be alleviated with multiple source frames, yet M2O splatting still suffers from stray effects at boundaries due to its image formation model that is less flexible than M2M splatting. Best viewed when zoomed in.

a many-to-one (M2O) propagation where the pixel set after fusion is smaller than the actual pixel set of frame. This results in holes as shown in Fig. 5 (a). Though exploiting multiple source frames lessens this issue, M2O warping still restricts each source pixel to only render a small 4-pixel vicinity in the output frame. This limits the effectiveness in representing and thus interpolating regions with complex interactions among the pixels, as shown in Fig. 5 (b).

Fortunately, such limitations can be alleviated through many-to-many (M2M) pixel splatting by using multiple motion vectors to model the motion of each source pixel. We forward warp each pixel in the source s with N ($N > 1$) sub-motion vectors to t, and get the set of warped pixels,

$$\hat{I}_{s\to t} = \bigcup_{n=1}^{N} \hat{I}^{n}_{s\to t} \tag{2}$$

Many-to-many splatting relaxes the restriction that each source pixel can only contribute to a single location. Therefore it allows the underlying motion estimator to learn to reason about occlusions, and model complex color interactions across a larger area of pixels.

Pixel Fusion. By applying M2M warping to all the input pixels in $\{I_0, I_1\}$, we get the complete warped pixel set where multiple target pixels may correspond to the same pixel locations: $\hat{I}_t = \hat{I}_{0\to t} \bigcup \hat{I}_{1\to t}$. To fuse warped pixels overlap with each other, we measure each of the pixels' importance from three aspects: the temporal relevance, brightness consistency, and the reliability score.

1) Temporal Relevance $\mathbf{r}_i$ characterizes changes not based on motion (e.g. lighting changes) between a source frame and the target. For simplicity, we adopt linear interpolation by setting $r_i = 1 - t$ if i comes from I_0 and $r_i = t$ otherwise, with t being the desired interpolation time.

2) Brightness Consistency $\mathbf{b}_i$ indicates occlusions by comparing a frame to its target through backward warping:

$$b_i = \begin{cases} -1 \cdot ||I_0(i) - I_1(i + F_{0\to1}(i))||_1, & \text{if } i \in I_0, \\ -1 \cdot ||I_1(i) - I_0(i + F_{1\to0}(i))||_1, & \text{if } i \in I_1, \end{cases} \tag{3}$$

The effectiveness of Eq. 3 is not decided only by the motion but also by the pixels' colors, which can be affected by various factors like noise, ambiguous appearance, and changes in shading [1, 28]. To enhance the robustness, we thus further adopt a learned per-pixel reliability score.

3) Reliability Score $\mathbf{s}_i$ is jointly estimated together with the motion vectors through the Motion Refinement Network as introduced in Sec. 3.1 and learned from data.

With these three measurements, we fuse the overlapped pixels at a location j in the form of weighted summation,

$$I_t(j) = \frac{\sum_{i \in \hat{I}_t} \mathbb{1}_{i=j} \cdot e^{(b_i \cdot s_i \cdot \alpha)} \cdot r_i \cdot c_i}{\sum_{i \in \hat{I}_t} \mathbb{1}_{i=j} \cdot e^{(b_i \cdot s_i \cdot \alpha)} \cdot r_i} \tag{4}$$

where c_i represents the i-th warped pixel's original color, α is a learnable parameter adjusting the scale of weights, $\hat{I}_t$ is the set of all the warped pixels at time t, and $\mathbb{1}_{i=j}$ indicates if the warped pixel i is mapping to the pixel location j.

We note that our final fusion function is similar to SoftSplat [28] in the form of softmax weighting, however our method differs in three aspects. First, we provide a solution to directly operate in the pixel color domain, while SoftSplat splats features and utilizes an image synthesis network instead. Second, we propose a general framework for fusing pixels from multiple frame, while SoftSplat fuses each frame individually. Third, we introduce the learning based reliability score to fuse overlapping pixels in a data-driven manner while SoftSplat uses feature consistency.

4. Experiments

In the section, we subsequently compare our proposed proposed to related state-of-the-art frame interpolation techniques and analyze it quantitatively as well as qualitatively.

4.1. Datasets

We supervise our proposed approach on the training split of Vimeo90K and test it on various datasets summarized as follows: 1) Vimeo90K [46], the test split containing 3,782 triplets at a resolution of 448×256 pixels. 2) UCF101 [41],

	GFLOPs		Speed ms/f	Arbitrary Interp.	Vimeo90K		UCF101		ATD12K		Xiph-2k		Xiph-“4k”	
	share	unshare			PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
SepConv [30]	N/A	93	101		33.79	.970	34.78	.967	27.40	.950	34.77	.929	32.06	.880
DAIN [2]	712	1308	977	✓	34.71	.976	35.00	.968	27.38	.955	35.97	.940	33.51	.898
CAIN [7]	N/A	29	47		34.65	.973	34.98	.969	25.28	.952	35.21	.937	32.56	.901
AdaCoF [16]	N/A	117	36		34.47	.973	34.90	.968	27.75	.950	34.82	.927	32.19	.882
SoftSplat [28]	95	218	122	✓	36.10	.980	**35.39**	.970	28.22	.957	**36.62**	.944	33.60	.901
BMBC [32]	441	376	1213	✓	35.01	.976	35.15	.969	27.68	.945	–	–	–	–
RIFE [11]	N/A	20	**17**		35.51	.978	35.25	.969	28.59	.953	36.15	.962	33.27	.942
ABME [33]	N/A	549	497		**36.18**	**.981**	35.38	.970	28.71	**.959**	35.18	.964	32.36	.940
M2M-PWC	87	< 1	32	✓	35.40	.978	35.17	**.970**	**29.03**	**.959**	36.45	**.967**	**33.93**	**.945**
M2M-DIS	**61**	< 1	28	✓	35.06	.976	35.13	.968	28.95	.956	36.14	.965	33.25	.942

Table 1. Quantitative results on the Vimeo90K, UCF101, ATD12K, and Xiph datasets. We compute models' GFLOPs and speed based on 640×480 inputs. The “share” denotes the part of compute independent from the desired frame rate, which is in contrast to “unshare”.

a dataset containing human action videos of size 256×256 pixels. A set of 379 triplets were selected by Liu *et al.* [19] as a test set for frame interpolation. 3) Xiph [25], as proposed by Niklaus *et al.* [28] where “Xiph-2K” is generated by downsampling 4K footage, and “Xiph-4k” is based on center-cropped 2K patches. 4) ATD12K [17], containing 2,000 triplets from various animation videos at a resolution of 960×480 pixels. 5) X-TEST [39], the test set from X4K1000FPS [39], containing 15 scenes extracted from 4K videos at 1000fps. We denote the original resolution as X-TEST(4K), and additionally adopt X-TEST(2K) by downsampling X-TEST(4K) by a factor of two.

4.2. Training

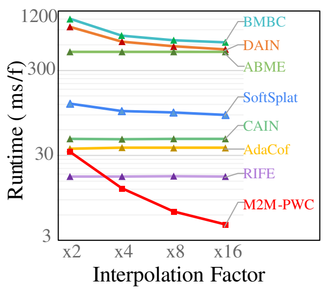
We train our proposed pipeline in an end-to-end manner. Given an output I_t and the ground truth I_t^{gt}, we define the training loss as the sum of the Charbonnier loss [4] and the census loss [21], $L = L_{char} + L_{cen}$. To train the model, we utilize the 51,312 triplets from the training split of Vimeo90K [46]. We apply random data augmentations including spatial and temporal flipping, color jittering, and random cropping with 256×256 patches. We adopt Adam [20] for optimization, with a weight decay of 1e-4. We train the model for 400k iterations with a batch size of 8, during which the learning rate is decayed from 1e-4 to 0 via cosine annealing. All experiments are implemented with PyTorch, and executed on a single Nvidia Titan X.

4.3. Comparison with State-of-the-art

We report two variants of our proposed approach based on different methods for estimating the off-the-shelf motion vectors. “M2M-PWC” is based on PWC-Net [42]. In this setting, we jointly optimize PWC-Net during training and generate initial flows at 1/4 of the original resolution. The other variant is based on DISFlow [15] and denoted as “M2M-DIS”. In our experiments, we generate N=4 sub-motion vectors for each pixel. For comparisons, we report the performance of recent VFI approaches including: SepConv [30], DAIN [2] CAIN [7], AdaCoF [16], SoftSplat [28], BMBC [32], RIFE [11], and ABME [33].

We first analyze the computational costs of these models in Tab. 1. We denote the required compute that is independent from the desired frame rate as “share”, and “unshare” otherwise. Hence the total computational complexity for interpolating n frames can be calculated through “#share+$n \cdot$ #unshare”. Motion-free methods (including SepConv, CAIN, and AdaCof) and pure bilateral-motion-based methods (like RIFE and ABME) have no share compute (denoted as “N/A”) and their computational complexity increases linearly in the number of desired frames. Approaches like SoftSplat, and BMBC can interpolate arbitrary frames, yet still suffer from both high compute and $unshare$ compute. E.g. in the $\times 8$ interpolation setting, they take 1.6 TFLOPs, and 3.1 TFLOPs respectively. In contrast, our M2M takes only 0.1 TFLOPs in total. Fig. 6 (a) compares the average runtime for different methods subject to varying interpolation factors. Our method is faster than all other methods in multi-frame settings. For $\times 16$ interpolation our method takes about 5 ms to interpolate a frame, which is around 5×, 20×, and 100× faster than RIFE, SoftSplat, and ABME respectively.

Taking efficiency aside, our method achieves state-of-the-art performance on multiple datasets. The metrics for $\times 2$ interpolation are presented in Tab. 1. On Vimeo90K and UCF101, our M2M method is on par with the recently proposed real-time method RIFE and performs slightly worse than SoftSplat and ABME. On Xiph-2K, our M2M method achieves slightly lower PSNR than SoftSplat, yet achieves the highest SSIM among all the methods. Moreover, on the animation dataset ATD12K and the high-resolution dataset Xiph-“4K”, our M2M method, especially M2M-PWC, outperforms previous methods in terms of both PSNR and SSIM. This demonstrates our methods' effectiveness when processing high-resolution videos and the ability to gener-



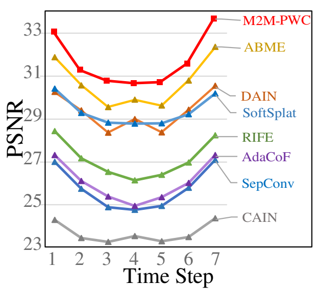


Figure 6. Evaluating multi-frame interpolation. (a) Runtime in logarithmic scale for interpolating 640×480 video frames with different interpolation factors. (b) Per-frame accuracy for ×8 interpolation on X-TEXT(2K). Best viewed in color.

	X-TEST(4K)		X-TEST(2K)		Runtime
	PSNR	SSIM	PSNR	SSIM	(ms/f)
SepConv [30]	23.94	.794	25.70	.800	693
DAIN [2]	26.78*	.807*	29.33	.910	3132
CAIN [7]	22.51	.775	23.62	.773	287
AdaCoF [16]	23.90	.727	26.03	.778	234
SoftSplat [28]	25.48	.725	29.73	.824	318
RIFE [11]	24.67	.797	27.49	.806	104
ABME [33]	30.16*	.879*	30.65	.912	2904
XVFI† [39]	30.12	.870	30.85	.913	203
M2M-PWC	**30.81**	**.912**	**32.07**	**0.923**	44
M2M-DIS	30.18	.909	30.98	0.912	**39**

Table 2. Quantitative results for ×8 interpolation on the X-TEST dataset. † indicates model trained with X-TRAIN. * indicates the numbers are copied from [33]. All the run-times are measured on X-TEST(2K).

alize across domains such as animation videos.

We report the results for ×8 interpolation on the X-TEST dataset, which contains diverse sequences with both high resolution and high frame rate, in Tab. 2. Our M2M method outperforms all previous methods on both the original 4K full resolution (4096×2160) and the downsampled 2K resolution (2048×1080) with substantial advantages in efficiency. For the models trained with Vimeo90K, ABME achieves the second-best PSNR in both 4K and 2K settings, but it takes 2,904ms to interpolate a 2K frame which is nearly 70× slower than M2M. To evaluate the temporal consistency, we compare the accuracy at each interpolation time step in Fig. 6 (b). We found that previous methods tend to deteriorate when interpolating frames that are temporally centered between the inputs, while M2M achieves a flatter and smoother curve for intermediate frames. This shows that M2M interpolates frames with not only better quality, but also higher temporal consistency.

4.4. Method Analysis

Ablation of Modules. We first analyze the effectiveness of the different components of our method in Tab. 3. We start with a single motion vector for each pixel. The first row demonstrates that directly using the off-the-shelf flow for warping leads to sub-optimal accuracy. As shown in the second row, applying the refinement network without joint flow encoding (JFE) and low-rank feature modulation (LFM) can already significantly improve performance by 0.97 dB and 2.38 dB for PWC-Net and DISFlow respectively. Further applying either JFE or LFM leads to improvements of more than 0.15 dB for both off-the-shelf flow methods. And using both JFE and LFM helps to boost the performance to 35.15 dB and 34.78 dB, respectively. In the last two rows, we also show the impact of the reliability scores which are generated by the refinement network and utilized for the pixel fusion. Without this score, the performance degrades, thus highlighting the importance of this metric in comparison to only using photoconsistency.

MRN	JFE	LFM	RS	PWC-Net	DISFlow
				33.97	31.93
✓				34.94	34.32
✓	✓			35.09	34.59
✓		✓		35.07	34.51
✓	✓	✓		35.15	34.78
✓	✓	✓	✓	35.24	34.93

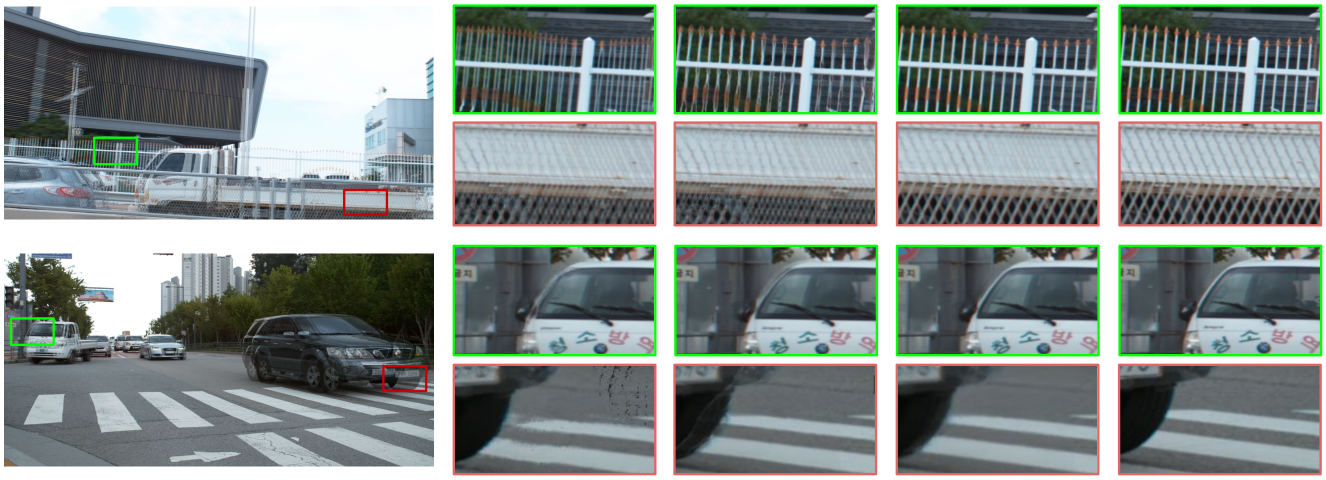
Table 3. Ablative experiments (in PSNR) on Vimeo90K with different initial flow methods. "MRN" denotes the motion refinement network, "JFE" refers to the joint flow encoding module in MRN, "LFM" is the low-rank feature modulation, and "RS" denotes the reliability score in the fusion step that synthesizes the output.

		N=1	N=2	N=4	N=8
PWC-Net	PSNR	35.24	35.35	35.40	35.39
	Runtime	16	16	17	20
DISFlow	PSNR	34.93	34.98	35.06	35.07
	Runtime	12	12	13	15

Table 4. Analyzing the impact of the number of sub-motion vectors for each pixel in our many-to-many splatting on Vimeo90K, with two different initial flow estimators.

Effect of Number of Flows per Pixel. Tab. 4 compares the effect of using different numbers of the sub-motion vectors for the M2M splatting. When N=1, it reduces the warping to M2O mapping, and achieves the lowest accuracy. When increasing N to 4, M2M improves the accuracy by more than 0.1 dB, with a very slight increment in run-time (<1ms). Also, and as shown in the last row, we noticed that further increasing the number of sub-motion vectors leads to marginal improvements. Fig. 7 illustrates the visual results for M2O splatting and M2M splatting.

Effect of Resolution for Initial Flow Estimation Our method relies on an off-the-shelf optical flow estimator to generate the initial flow. However, most optical flow estimation models are trained using a relatively low resolutions. Directly applying them to estimate the flow at 2K or 4K in-



(a) Overlapped inputs (b) Initial flow (c) N=1 (d) N=4 (e) Ground-truth

Figure 7. Comparison between many-to-one splatting and many-to-many splatting. Given the input frames (a), M2O splatting with initial flow (b) or single refined sub-motion vector (c) results in undesired visual artifacts for regions with complex motion. In comparison, our proposed M2M splatting with four sub-motion vectors (b) can interpolate with much higher quality.

	R=	Xiph-2K	Xiph-"4k"	X-TEST(2K)	X-TEST(4K)
PWC-Net	1	36.15	32.94	28.35	24.85
	2	**36.45**	33.76	31.00	27.08
	4	36.36	**33.93**	**32.07**	29.65
	8	35.74	33.75	31.65	**30.81**
DISFlow	1	**36.14**	**33.25**	31.03	**30.18**
	2	36.05	33.18	**31.18**	30.06
	4	35.73	32.94	30.54	29.68
	8	35.13	32.29	29.49	28.66

Table 5. Impact of the resolution at which the initial optical flow estimator is applied on. "R" is the down-sampling factor.

puts may result in sub-optimal results. We thus study the impact of the initial flow's resolution for interpolating high-resolution frames in Tab. 5. Since PWC-Net is learning-based and pre-trained on small resolutions, it is less effective at processing high-resolution frames as demonstrated by the reduced interpolation quality on 4K data. By down-sampling the input by a factor of 4 or 8, the accuracy improves significantly. In contrast, DISFlow is not supervised and hence less susceptible to similar domain gaps.

Discussions and Limitations. Though our method achieves very high efficiency especially for high framerate interpolation, its accuracy on low-resolution datasets like Vimeo90K is behind several state-of-the-art methods. We believe that carefully tuning and enlarging the model capacity allows M2M to compete with these state-of-the-art methods. The proposed method renders intermediate frames based on forward warping, which may be subject to holes in the output. In Fig. 8, we count the average number of remaining holes (in pixels) for different configurations on Vimeo90K. As we can see, our M2M splatting with N=4 is still subject to around 0.5-pixel holes in each frame on average. However, compared to the initial single sub-motion based M2O splatting, our method has significantly decreased the number of holes. Another limitation of our method is that the many-to-many splatting process may result in blurriness as shown in Fig. 7 (d). This can be addressed by further improving the fusion strategy or applying a lightweight network to refine the output.

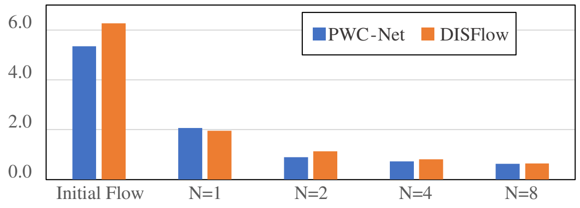


Figure 8. Analysis of number of remaining holes (in pixels) versus the number of sub-motion vectors in many-to-many splatting.

5. Conclusion

In this work, we present a many-to-many splatting technique to efficiently interpolate intermediate video frames. We first design a motion refinement network to generate multiple sub-motion vectors for each pixel. These sub-motion fields are then applied to forward warp the pixels to any desired time step, which are then fused to obtain the final output. By sharing the computation for the flow refinement and only requiring little compute to generate each frame, our method is especially well-suited for multi-frame interpolation. Experiments on multiple benchmark datasets demonstrate that the proposed method achieves effectiveness with superior efficiency.

Acknowledgements.Acknowledgements. This work was supported in part by DARPA and NSF.

References

[1] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *Int. J. Comput. Vis.*, 92(1):1–31, 2011. 1, 2, 5

[2] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *CVPR*, 2019. 1, 2, 6, 7

[3] Wenbo Bao, Wei-Sheng Lai, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019. 2

[4] Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *ICIP*, 1994. 6

[5] Wanli Chen, Xinge Zhu, Ruoqi Sun, Junjun He, Ruiyu Li, Xiaoyong Shen, and Bei Yu. Tensor low-rank reconstruction for semantic segmentation. In *ECCV*, 2020. 4

[6] Xianhang Cheng and Zhenzhong Chen. Video frame interpolation via deformable separable convolution. In *AAAI*, 2020. 1, 3

[7] Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. Channel attention is all you need for video frame interpolation. In *AAAI*, 2020. 1, 3, 6, 7

[8] Myungsub Choi, Suyoung Lee, Heewon Kim, and Kyoung Mu Lee. Motion-aware dynamic architecture for efficient frame interpolation. In *ICCV*, 2021. 1, 3

[9] Tianyu Ding, Luming Liang, Zhihui Zhu, and Ilya Zharkov. Cdfi: Compression-driven network design for frame interpolation. In *CVPR*, 2021. 1, 3

[10] Weisheng Dong, Guangming Shi, Xiaocheng Hu, and Yi Ma. Nonlocal sparse and low-rank regularization for optical flow estimation. *IEEE Trans. Image Process.*, 23(10):4527–4538, 2014. 4

[11] 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*, 2020. 1, 2, 6, 7

[12] 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 *CVPR*, 2018. 1, 2

[13] Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran. Flavr: Flow-agnostic video representations for fast frame interpolation. *arXiv preprint arXiv:2012.08512*, 2020. 1, 3

[14] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009. 4

[15] Till Kroeger, Radu Timofte, Dengxin Dai, and Luc Van Gool. Fast optical flow using dense inverse search. In *ECCV*, 2016. 1, 2, 3, 6

[16] Hyeongmin Lee, Taeoh Kim, Tae-young Chung, Daehyun Pak, Yuseok Ban, and Sangyoun Lee. Adacof: Adaptive collaboration of flows for video frame interpolation. In *CVPR*, 2020. 1, 3, 6, 7

[17] Haopeng Li, Yuan Yuan, and Qi Wang. Video frame interpolation via residue refinement. In *ICASSP*. IEEE. 6

[18] Yihao Liu, Liangbin Xie, Li Siyao, Wenxiu Sun, Yu Qiao, and Chao Dong. Enhanced quadratic video interpolation. In *ECCV*, 2020. 1

[19] Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *ICCV*, 2017. 2, 6

[20] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018. 6

[21] Simon Meister, Junhwa Hur, and Stefan Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, 2018. 6

[22] Simone Meyer, Victor Cornillère, Abdelaziz Djelouah, Christopher Schroers, and Markus Gross. Deep video color propagation. *BMVC*, 2018. 1

[23] Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus Gross, and Christopher Schroers. Phasenet for video frame interpolation. In *CVPR*, 2018. 1

[24] Simone Meyer, Oliver Wang, Henning Zimmer, Max Grosse, and Alexander Sorkine-Hornung. Phase-based frame interpolation for video. In *CVPR*, 2015. 1

[25] Christopher Montgomery. Xiph.org video test media (derf's collection). In *Online,https://media.xiph.org/video/derf/*, 1994. 6

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

[27] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *CVPR*, 2018. 1, 2

[28] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *CVPR*, 2020. 1, 2, 5, 6, 7

[29] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *CVPR*, 2017. 2

[30] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *ICCV*, 2017. 1, 3, 6, 7

[31] Simon Niklaus, Long Mai, and Oliver Wang. Revisiting adaptive convolutions for video frame interpolation. In *WACV*, 2021. 1, 3

[32] Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. In *ECCV*, 2020. 1, 2, 6

[33] Junheum Park, Chul Lee, and Chang-Su Kim. Asymmetric bilateral motion estimation for video frame interpolation. In *ICCV*, 2021. 1, 2, 6, 7

[34] Tomer Peleg, Pablo Szekely, Doron Sabo, and Omry Sendik. Im-net for high resolution video frame interpolation. In *CVPR*, 2019. 1, 2

[35] Anh-Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavskỳ, Valeriy Glukhov, Ivan Oseledets, and Andrzej Cichocki. Stable low-rank tensor decomposition for compression of convolutional neural network. In *ECCV*, 2020. 4

[36] Fitsum A Reda, Deqing Sun, Aysegul Dundar, Mohammad Shoeybi, Guilin Liu, Kevin J Shih, Andrew Tao, Jan Kautz,

and Bryan Catanzaro. Unsupervised video interpolation using cycle consistency. In *ICCV*, 2019. 1

[37] Richard Roberts, Christian Potthast, and Frank Dellaert. Learning general optical flow subspaces for egomotion estimation and detection of motion anomalies. In *CVPR*, 2009. 4

[38] Laura Sevilla-Lara, Deqing Sun, Varun Jampani, and Michael J. Black. Optical flow with semantic segmentation and localized layers. In *CVPR*, 2016. 4

[39] Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. Xvfi: Extreme video frame interpolation. In *ICCV*, 2021. 1, 2, 6, 7

[40] Li Siyao, Shiyu Zhao, Weijiang Yu, Wenxiu Sun, Dimitris Metaxas, Chen Change Loy, and Ziwei Liu. Deep animation video interpolation in the wild. In *CVPR*, 2021. 1

[41] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 5

[42] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 1, 2, 3, 6

[43] Chengzhou Tang, Lu Yuan, and Ping Tan. Lsm: Learning subspace minimization for low-level vision. In *CVPR*, 2020. 4

[44] Stepan Tulyakov, Daniel Gehrig, Stamatios Georgoulis, Julius Erbach, Mathias Gehrig, Yuanyou Li, and Davide Scaramuzza. Time lens: Event-based video frame interpolation. In *CVPR*, 2021. 1

[45] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *ECCV*, 2018. 1

[46] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *Int. J. Comput. Vis.*, 127(8):1106–1125, 2019. 1, 2, 5, 6

[47] Zhiyang Yu, Yu Zhang, Deyuan Liu, Dongqing Zou, Xijun Chen, Yebin Liu, and Jimmy S. Ren. Training weakly supervised video frame interpolation with events. In *ICCV*, pages 14589–14598. 1

[48] Liangzhe Yuan, Yibo Chen, Hantian Liu, Tao Kong, and Jianbo Shi. Zoom-in-to-check: Boosting video interpolation via instance-level discrimination. In *CVPR*, 2019. 2

[49] Haoxian Zhang, Yang Zhao, and Ronggang Wang. A flexible recurrent residual pyramid network for video frame interpolation. In *ECCV*, 2020. 2

[50] Shipeng Zhang, Lizhi Wang, Lei Zhang, and Hua Huang. Learning tensor low-rank prior for hyperspectral image reconstruction. In *CVPR*, 2021. 4

[51] Youjian Zhang, Chaoyue Wang, and Dacheng Tao. Video frame interpolation without temporal priors. *NeurIPS*, 2020. 1