Multi-Level Adaptive Separable Convolution for Large-Motion Video Frame Interpolation

Ruth Wijma
University of Amsterdam
rlwijma@gmail.com

Shaodi You
University of Amsterdam
s.you@uva.nl

Yu Li
Applied Research Center (ARC), Tencent PCG
ianyli@tencent.com

Abstract

Current state-of-the-art methods within Video Frame Interpolation (VI) fail at synthesizing interpolated frames in certain problem areas, such as when the video contains large motion. This work aims at improving performance on frame sequences containing large displacements by extending the Adaptive Separable Convolution model in two ways. First of all, we increase the receptive field of the model by utilizing spatial pyramids, which efficiently increase the interpolation kernel size. We additionally adapt the network to accommodate for four frames, as opposed to just two, which should give it the ability to learn more complex motion patterns. This work also introduces the Large-Motion Video Interpolation Dataset (LMD), which contains extracted frames from videos containing large displacements and highly non-linear movements. Our analysis shows that applying the model changes, together with the use of our new dataset, does indeed result in improved performance on large displacement videos. We also show that the increase in performance generalizes to frame sequences of all sorts by outperforming other models in our benchmark on most tasks, and almost setting the new state-of-the-art on the Vimeo-90K dataset.

1. Introduction

Video Frame Interpolation (VI) is one of the prominent fields within computer vision, one that has been the focus of many researchers. The process of VI is the insertion of generated frames in between existing frames within a video, with the goal of improving its fluency and visual appeal. These methods are helpful in cases where we have a video with a frame rate that is too low to be used in practice. For example, movies are often shot in 24 frames per second (FPS), but the FPS needs to be up-scaled to 60FPS before the video can be shown on a modern TV. Another field that uses VI methods is the medical sector [14]. Computed Tomography (CT) scans generally have a very low frame rate, because of the high radiation emitted per scan. Medical practitioners would however benefit from having more frames to make more informed decisions [4].

However, many of the state-of-the-art approaches suffer from artifacts in the synthesized images as a result of large frame-to-frame pixel-wise motion being present in the frame sequence, meaning that objects within the video either move fast in space, the video resolution is large or that the frame rate is relatively low. We either see the same object in different places, or the resulting frame contains a large amount of blur, *i.e.* a lack of detail. In movies, the occurrence of these artifacts even has a specific name, called the Soap Opera effect.

In summary, this work will address these problems by extending the Adaptive Separable Convolution model [11] to work with more input frames and increase the receptive field by utilizing spatial pyramids to efficiently make use of the interpolation kernels. We also introduce a method that specifies a notion of displacement of a frame sequence by measuring the frame-to-frame flow vectors. This allows us to test model performance as a function of the estimated displacement.

This work also introduces the Large-Motion Video Interpolation Dataset (LMD) consisting of many frame sequences extracted from videos of extreme sports, therefore consisting of many large displacements and non-linear movements. The usage of this dataset is twofold, we first of all include the train fold of the LMD as a subset of our training dataset, while we utilize another subset of the LMD as validation data.

We will conclude by comparing several recent methods on multiple datasets and metrics to show that the incorporation of the aforementioned changes leads to improved interpolated results. Additionally, we do a qualitative analysis by showing the synthesized images on some hard frame sequences. Our experiments show that we are able to improve upon the problem areas mentioned above. We also see a significant improvement in general performance, as

measured on the Vimeo-90K test set. Code can be found at https://github.com/rwq/MSC_AI_Thesis_FI

We make the following contributions:

- We introduce our extension of the Adaptive Separable Convolution network, named Multi-Level Adaptive Separable Convolution.
- We introduce our dataset containing mostly frame sequences having larger displacements, The Large Motion VI Dataset.
- We show that our model is able to improve upon frame sequences containing large displacements, as well as improve upon general performance, as measured on the Vimeo-90K test set.

2. Related Work

Most Video Frame Interpolation methods can be categorized as being either flow-based or kernel-based. While the former first estimates the flow vectors between the input frames as an intermediate step, and uses this to synthesize the output, models of the latter category do this directly.

Xue *et al.* proposed Task-Oriented FLow (TOFlow) [19], an approach that first estimates optical flow between frames, and then uses this flow in conjunction with the two frames to generate intermediate frames. It gets its name from the fact that they train a network based on of three tasks: Temporal frame interpolation, Video denoising/deblocking and Video super-resolution. A later work from Bao *et al.* also noticed that occlusion is a problem with regards to increasing the interpolation performance [1]. They proposed Depth-Aware video frame INterpolation (DAIN), in which they use the fact that objects that are closer to the camera tend to have a larger motion (larger flow vectors), and should thus be paid more attention to. In reconstructing the intermediate flow vectors, pixels that are closer are weighted more heavily, thereby decreasing the chance of sampling a pixel that is occluded in the intermediate frame. Xu *et al.* [18] proposed the first deep-learning method to make use of more than two frames during interpolation. Instead, they use four frames, *i.e.* two before and two after the interpolated image. The authors hypothesize that using more frames enables the model to learn motion more complex than linear motion, which is not possible from using just two frames. They report that their method outperforms other high-scoring methods.

Niklaus *et al.* were one of the first to successfully use a convolutional neural network (CNN) in solving the video interpolation problem [10]. In contrast to other methods that also use an encoder-decoder architecture [7], they do not directly output the predicted interpolated frame, but instead predict 41×41 kernels which are used to synthesize each pixel based on nearby pixels. These locally adaptive convolutional kernels are then convoluted with the two input frames to produce the predicted interpolated frame. Specifically, given two input frames, the network produces two kernels for each pixel in the output. These interpolation kernels are then convolved with the input frames to produce the interpolated frame $\hat{\mathbf{I}}$.

The same group of researchers made use of an interesting property of 2D kernels in a follow-up paper a few months later, where they used the fact that a 2D kernel can be approximated by using two 1D kernels [11]. This is true because the outer product of two 1D kernels is a 2D kernel. This allowed them to drastically lower the number of pixels per output interpolation kernel from n^2 to $2n$, with n being the kernel size. They mention that this approach is over 20 times faster on Full HD video. The memory needed to hold the output of a 1080p video in memory dropped to 1.27 GB. Since the complexity is now linear, it allowed them to increase the kernel size to 51×51, without a significant drop in performance.

3. Problem Statement

The problem of video frame interpolation can be described as follows: given some video sequence having n frames, $(\mathbf{I}_t)_{t=1}^{n}$, we want to create a new frame t, where $t \in (1, n)$. Each frame is represented as a 3D tensor of size $3 \times H \times W$, where H and W are the height and the width of the frame in pixels respectively. Each pixel is represented by three integer intensity values in $[0, 255]$, one for each of its three channels, which are red, green and blue. Generally, to interpolate a frame at time t', VI methods use nearby frames $\{\mathbf{I}_t : t' \neq t\}$ to synthesize the intermediary frame $\mathbf{I}_{t'}$.

Before we continue, we introduce our measure of displacement. This measure is based on the optical flow-vectors estimated by PWC-net [17]. Given an image pair $(\mathbf{I}_a, \mathbf{I}_b)$, represented by two of $3 \times H \times W$ tensors, this method estimates the flow vectors $F^{\rightarrow}$ from $\mathbf{I}_a$ to $\mathbf{I}_b$. These vectors are then represented by a tensor of shape $2 \times H \times W$, where the first dimension is used to store the flow-vector direction, *i.e.* the vertical or horizontal dimension.

We first of all estimate the flow from $\mathbf{I}_{T-1}$ towards $\mathbf{I}_{T+1}$, denoted by $F^{\rightarrow}$, and we similarly estimate the flow in the opposite direction, denoted by $F^{\leftarrow}$. We then take the mean over all dimensions after summing the L_1-norms of both tensors. The displacement D for a frame sequence is then given by

$$D = \frac{||F^{\rightarrow}||_1 + ||F^{\leftarrow}||_1}{4HW}. \tag{1}$$

The reason for measuring distance using the mean absolute value, and not using Euclidean distance, is that the absolute value is more representative as a measure of displacement. This reasoning is graphically illustrated in Figure 1. Whether a displacement of a particular magnitude is visible within a square perceptual field is only dependent on the absolute displacement along the vertical and horizontal dimension. A movement of a particular Euclidean distance

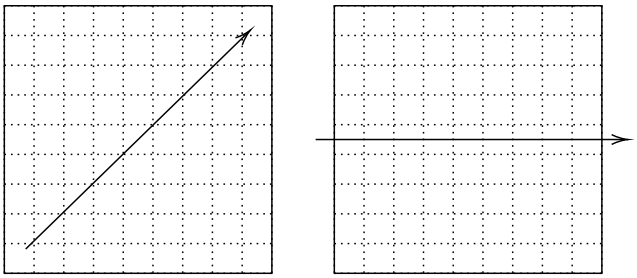

Figure 1: This figure shows the perceptual field of a VI model, as visualized by the squares. The frame-to-frame movement is visualized by the arrow. While both movements, left and right, have the same Euclidean distance, only the one on the left is within the field-of-view.

might be visible if it is placed diagonally on the perceptual field, but not if it perfectly follows the horizontal or vertical axis. In contrast, a movement measured using the mean absolute value will be visible irrespective of its orientation, given a constant length.

As stated earlier, the methods that are considered state-of-the-art suffer from some drawbacks, and one of these is large motion. Often, when large motion is present, the resulting synthesized interpolated frames contain artifacts. These artifacts are a result of the model not knowing how to make sense of the input, and as a result, the objects contained in the scene are either presented multiple times, or are simply blurred heavily.

We belief there are a variety of reasons for the occurrence of this problem. First of all, many of the methods specified in Section 2 are trained mostly on frame sequences containing small movements. For example, some of the large-scale datasets in VI, such as the Vimeo-90K dataset [19], only contain frame sequences with a mean flow lower than 8 pixels. When a model is trained on such a dataset, it is clear that it won't learn large displacements. Another possible explanation is that the models are simply not expressive enough. In order to model large displacement correctly, the interpolation mechanism needs to have the ability to find the pixel correspondence across the input frames. *E.g.* [10] and [11] use interpolation kernels of sizes 41×41 and 51×51 respectively. This means that every pixel in the synthesized frame is modeled as a function of the neighboring 20 and 25 pixels in either direction. It is then obvious that movements larger than this perceptive field cannot be learned by these models.

These are not the only areas in which current VI methods lack in performance. Additionally, many of the methods presented in Section 2 lack detail around edges, such as text and certain patterns or textures. We belief that this is a problem related to an overall lower performance of the model and not necessarily due to the network not being as expressive enough, or a lack of training data. All VI methods estimate optical flow, either implicitly, or explicitly as an intermediary step towards frame synthesis using a separate module. These optical flow mechanisms suffer when they cannot decide on the location of certain key points in the frames. The result is that the repeating pattern cannot be followed as there are too many matches. We belief this is is the reason for the occurrence of the artifacts when these patterns are present.

4. Large-Motion VI Dataset

One of the main contributions of this work is our newly created Large-Motion Video Interpolation Dataset (LMD). As the name suggests, this dataset mainly consists of videos that contain large frame-to-frame motion. The dataset contains videos on BMXing, football, wingsuit flying, skateboarding and other extreme sports. We also included several car and bike rides through cities such as California and Tokio, as their 4K resolution combined with a downsampled frame rate allowed us to collect high quality frame sequences containing large frame-to-frame movements. The variety in the scenes, the fast moving environment and the variety within the scenes all make for a complex and useful data source.

The problem with datasets in the VI literature is that they are insufficient, because the sequences they contain are too simple and/or not diverse enough. They are either of low quality and/or have small frame-to-frame displacement, implying that these datasets are not useful when modeling larger motion.

4.1. Dataset Creation

This dataset was created by selecting a set of 40 videos from YouTube and ensuring that there was enough diversity between them. We only selected videos with a minimum resolution of 1280×720. The videos were selected on the perceived degree of motion present in them. We split all 40 videos into a train, validation and test fold, containing 8.737, 1.435 and 1.509 sequences respectively, and made sure that each fold contains a similar set of videos.

The process of gathering videos for- and preparing the dataset involved several steps. First, we extracted all frames from the videos and randomly downsample the video from the original resolution to either 720p, 1080p or 1440p. This is another way to ensure that we get a wide range of displacements, since changing the resolution of a video will change the magnitude of all frame-to-frame displacements. We sample a frame sequence of 7 frames every 90 frames to prevent near-duplicates within the dataset. We then remove the second and sixth frame, since they are not needed during training, as stated in Section 3, resulting in 5 frames per sequence. In order to prevent scene-boundaries from being present in the frame sequences, we measure the SSIM be-

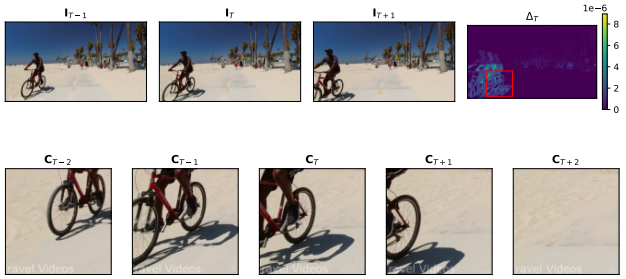

Figure 2: **(a)** A visual illustration of the weighted sampling procedure used during cropping. From left to right: the frame before the ground truth, $\mathbf{I}_{T-1}$, the ground truth, $\mathbf{I}_T$, and the frame after, $\mathbf{I}_{T+1}$. The fourth figure shows the normalized sampling weights Δ_T, a lighter color indicating a higher sampling probability. **(b)** The cropped frame sequence resulting from the weighted sampling procedure, denoted by $\mathbf{C}_i$.

tween all 5 frames and discard the sequence when one of the corresponding values falls below a certain threshold. Empirically, a value of 0.35 worked well, as it removed most scene boundaries, without resulting in any false positives. A small number of sequences containing scene boundaries were removed manually.

We additionally created a cropped version of the train and validation set, with a crop size of 256×256. To make sure that the cropped area is interesting, we use a weight matrix Δ_T, which is obtained by summing the squared L_2-norms of the image time derivatives, as specified by

$$\Delta_T = ||\mathbf{I}_T - \mathbf{I}_{T-1}||_2^2 + ||\mathbf{I}_T - \mathbf{I}_{T+1}||_2^2, \qquad (2)$$

where $\mathbf{I}_T$ is the ground truth frame intensity, and $\mathbf{I}_{T-1}$ and $\mathbf{I}_{T+1}$ are the preceding and succeeding frame intensities respectively. These intensities are computed by summing over the channels for each frame. We then use Δ_T as sampling weights for the center pixel of the crop. To ensure that the crop will be completely within the image, we set all weights within 128 pixels from the image borders to 0. We then use a Multinomial distribution to select the pixel, and crop the area around it. A graphical illustration is given in Figure 2, showing that this simple method is able to localize the motion present within the frame sequence.

4.2. Dataset Statistics

In order to quantitatively show the difference between the LMD and existing datasets, we compared the displacement and non-linearity with the Adobe240 and Vimeo-90K datasets, as shown in Figure 4. Both the displacement and non-linearity are much larger for our dataset, meaning that it is indeed useful for both training or validating a VI model with a focus on more complicated frame sequences. Figure 3 shows a few examples from the LMD and illustrates the non-linearity present within them.

5. Proposed Solution

This section will cover all the aspects of our proposed solution. We will start by introducing our extension to the Adaptive Separable Convolution (SepConv) model, which was originally proposed by Niklaus *et al.* [11]. We will then detail the procedures we utilize during training and specify the data augmentations steps we take.

5.1. Multi-Level Adaptive Separable Convolution

First of all, we chose to use the SepConv model to be the basis for our method, because of its favorable computational complexity compared to other methods, *e.g.* it is over 10 times faster than DAIN [1] during training and evaluation, while still having comparable performance. The computational efficiency allows us to do many experiments and simultaneously gives us the reassurance that any extension of our method would still be within the limitations of the GPUs we are using to train our models.

We extend the original SepConv architecture in two ways. We first of all increase the models ability to model larger motion in an efficient way by introducing a spatial pyramid. Secondly, we feed four frames into the model, instead of two, to make the model more expressive towards frame sequences containing non-linear movements.

We implement these changes as follows: we let each sub-network produce both the interpolation kernel at the original size and the displacement kernel. We do this by duplicating the sub-network structure after the first two convolutional and ReLU pairs. This means that we split the structure of the sub-network into two identical paths, each containing a separate convolutional layer with ReLU activation, followed by the upsampling and convolutional layer. The input for both paths is the output of the second ReLU layer and is thus identical. We chose not to predict all interpolation kernels using a separate sub-network to keep the increase in computational complexity to a minimum. Secondly, we also double the number of sub-networks from 4 to 8, to accommodate for the increase in the number of input frames.

Our method then estimates the interpolated frame $\hat{\mathbf{I}}$ first by computing the 2D interpolation kernels from their 1D counterparts, $k_{\cdot,h}$ and $k_{\cdot,v}$, using the matrix outer product. We then simply apply a local convolution, on each input frame with their respective interpolation kernels at the original scale and then sum this output with the interpolation result of our model on the downsampled version of the frame sequence, after upsampling it back to the original size. Downsampling with a factor s, denoted by $D_s(\cdot)$, and upsampling with a factor s, denoted by $U_s(\cdot)$, are implemented using 2D average pooling and bilinear upsampling respectively. We require s to be a non-negative power of 2 due to the padding strategy we use. See Equation 3, where $\dot{*}$ and $\otimes$ denote the local convolution and outer product operators respectively. A graphical illustration of our proposed

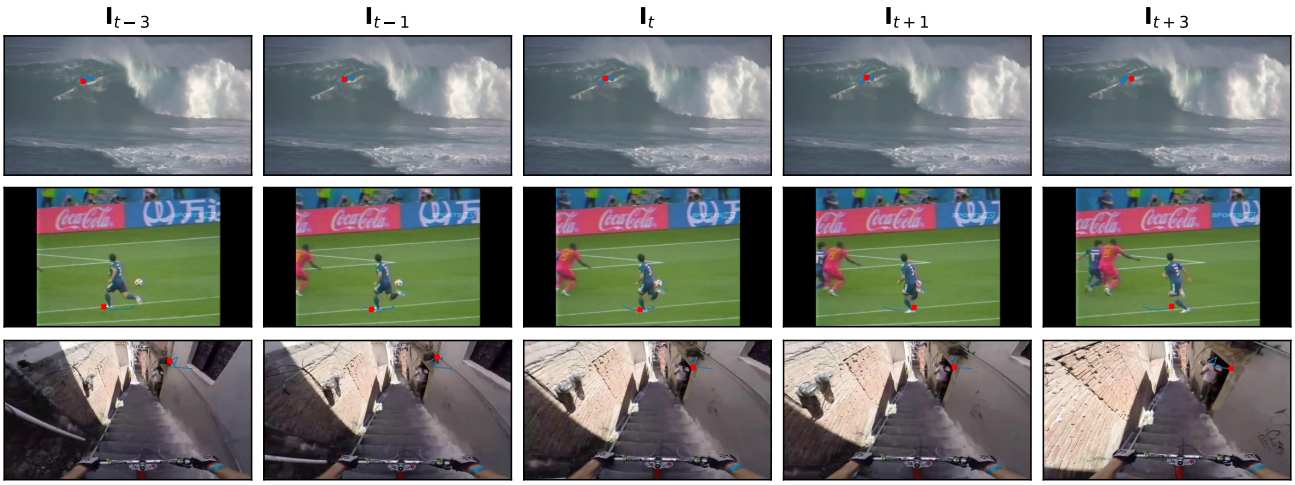

Figure 3: Example frame sequences extracted from the Large Motion Dataset (one sequence per row). Each frame sequence is accompanied by a tracked feature as shown in red, showing for each sequence the highly rigid and non-linear motion present in the videos. The trajectory across the sequence is shown in blue. The second sequence shows the tracking of the shoe of the football player wearing the blue jersey. Using only frames $\mathbf{I}_{t-1}$ and $\mathbf{I}_{t+1}$, *i.e.* using a linear model, would result in a poor approximation, given the quadratic curvature of this movement. Similar non-linear movements are ubiquitous in the LMD.

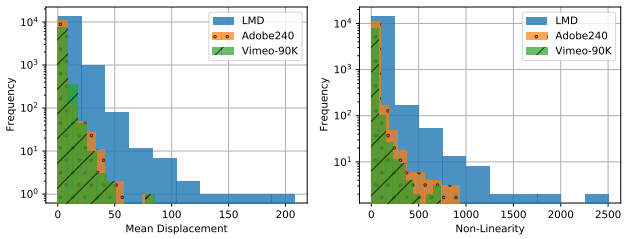

Figure 4: Two log-scale histograms showing the difference in the occurrence of harder to model frame sequences. We see that when we measure this by the mean displacement and non-linearity of a frame sequence, the LMD is much more complex when compared to both the Adobe240 and Vimeo-90K dataset. These measurements were taken over all folds. Non-linearity is measured as the mean squared normalized distance between the key-points in the frame and their linear approximation.

model architecture can be seen in Figure 5.

$$\hat{\mathbf{I}} = \sum_{i=1}^{4} \mathbf{I}_i \dot{*} (k_{i,h} \otimes k_{i,v}) + U_{s_i} \left[D_{s_i} (\mathbf{I}_i) \dot{*} (k^d_{i,h} \otimes k^d_{i,v}) \right] \tag{3}$$

5.2. Training Procedure

We will be doing all our experiments using either the Adaptive Separable Convolution model (SepConv) [11] or the extension we introduced in Section 5.1. We will make use of the publicly available $\mathcal{L}_1$- or $\mathcal{L}_f$-weights, depending on whether we are training a model for quantitative or qualitative analysis. We will only load the pretrained weights to the encoder part of the network, *i.e.* the first four convolutional blocks up until the bottleneck, see Figure 5. This is done to reduce training time, while still preventing the introduction of any bias regarding the size of the interpolation kernels, which we will be changing. We will adjust the first convolutional layer to accommodate for the extra 2 input frames, and randomly initialize these extra weights.

We use the train fold of the Vimeo-90K dataset from which we use 90% and 10% to create our train and validation folds respectively. We additionally use the train fold of the LMD and remove all frame sequences having a mean displacement, as defined by Equation 1, lower than 6 pixels. We found that this value removed most sequences having nearly no movements. We combine both datasets and utilize a sampling strategy that uses the normalized mean displacement as sampling weight. Each epoch we sample 50K samples without replacement from a total of approximately 64K samples. We randomly shuffle the training data and sample mini-batches with a batch size of 8.

We randomly flip the image sequence across its dimensions, *i.e.* horizontal, vertical and temporal, all with probability 0.5. This ensures that the model does not learn any biases with respect to the direction of movement in the training data. We also incorporate some augmentations that change the color schemes or brightness of the image. We apply Contrast Limited Adaptive Histogram Equalization (CLAHE), which is a method that reduces the differences

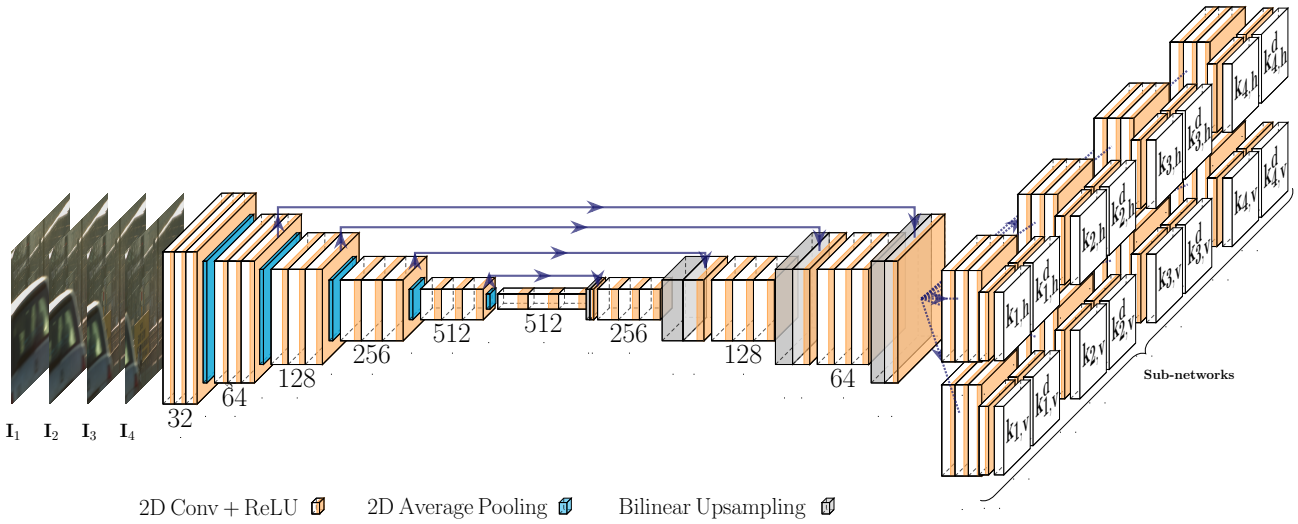

Figure 5: Our proposed extension of SepConv. Skip-connections are represented by purple lines. We extend the first layer to handle four input frames and change each sub-network so they output two 1D kernels, as opposed to just one. We additionally double the number of sub-networks to handle the doubling in the number of input frames. As can be seen from the figure, each *column* produces two 1D kernels, which can be combined to produce a 2D kernel via the application of the outer product.

in contrast across an image. We pair this with a method that randomly changes the brightness and contrast. This transformation makes sure that the model learns to interpolate frame sequences for any kind of lighting. We also use a method that changes the hue of the image. This ensures that we make the learning more uniform with respect to objects of different color. We finally add some Gaussian noise. These transformations are done with probability 0.2.

To speed up training, we randomly crop a section of the original frame sequence. This additionally acts as a way to combat overfitting, since it is less likely that the model learns the precise movements in the training data as it gets to see only a part of the frame each time it is sampled.

As stated above, we use two different loss functions to obtain our two models. We first of all train our model architecture by minimizing L_1-loss to obtain our $\mathcal{L}_1$-model. This loss function is defined as

$$L_1(\hat{\mathbf{I}}, \mathbf{I}) = \sum_{c,x,y} | \hat{\mathbf{I}}_{c,x,y} - \mathbf{I}_{c,x,y} | = ||\hat{\mathbf{I}} - \mathbf{I}||_1. \quad (4)$$

We similarly train our model on perceptual loss to obtain our $\mathcal{L}_f$-model. This loss function is defined by

$$L_f = L_1(\hat{\mathbf{I}}, \mathbf{I}) + ||\phi(\hat{\mathbf{I}}) - \phi(\mathbf{I})||_2^2, \quad (5)$$

where ϕ is the output of the relu4_4 layer of the VGG-19 network [15], similar to [11], and $\mathbf{I}$ and $\hat{\mathbf{I}}$ are the ground truth and synthesized interpolated frames respectively.

We utilize hyper-parameter tuning using the Tree-Structured Parzen Estimator [3] implemented in HyperOpt [2]. We optimize the crop size, the main kernel size and the choice of optimizer, where we evaluated both the Rectified Adam [6] and Ranger [12] optimizers.

5.3. Learning Rate Scheduling

We use two separate learning rates for the decoder and encoder part of the network, since they differ in their level of optimality as one is pre-trained and the other contains randomly initialized weights. We do this since we only want to fine-tune the encoder using the training data, while we still need to learn good parameter values for the decoder and the sub-networks. We set the learning rate of the encoder to 0.0001, while we use 0.001 for the other parts of the model. These values were high enough to elicit training, while preventing the introduction a large variance in the weight updates.

We noticed that using a flat learning rate was insufficient when training our models. Instead, we found that using a flat learning rate of 10 epochs, followed by a Cosine Annealing learning rate schedule [8] worked well to further improve our results. We slightly reduce both learning rates after 10 epochs towards $1e^{-5}$. We additionally found that *restarting* the training procedure helped to further improve our performance, therefore, we reset the learning rates back to their original values every 50 epochs.

5.4. Implementation Details

PyTorch [13] was used in combination with CUDA to speed up the training process by utilizing several Nvidia GPUs. We either used a GTX 1070 Ti, GTX TitanX or GTX 1080 Ti, depending on which GPU cluster we used, or if

Method	Vimeo-90K - D^{90} [19]			Vimeo-90K - Q^{90} [19]		
	PSNR	IE	SSIM	PSNR	IE	SSIM
DAIN [1]	31.235	8.054	0.915	32.558	7.119	0.934
QVI-lin [18]	29.572	9.463	0.889	29.232	9.832	0.891
QVI-quad [18]	**32.221**	**7.117**	**0.923**	32.332	6.997	0.933
SepConv-$\mathcal{L}_1$ [10]	30.349	8.788	0.901	31.671	7.684	0.924
Ours-$\mathcal{L}_1$	31.374	7.882	0.907	**32.749**	**6.868**	**0.930**

Table 1: Quantitative comparison on the complex subsets of the Vimeo-90K dataset.

we trained locally. Random seeds were used throughout all code to ensure reproducibility. Since the original implementation of SepConv does not include the backward pass of the custom CUDA layer, we resorted to using a different implementation, which can be found at https://github.com/HyeongminLEE/pytorch-sepconv, and use this to make our extended version of SepConv. We verified that both implementations produce the same output.

6. Experiments

6.1. Quantitative Analysis

Performance on Complex Frame Sequences We first test whether we solved the problem of the lower performance on frame sequences containing either large displacements and/or non-linear movements. We hypothesized that the implementation of the displacement interpolation kernels will increase the effective perceptual field of the model, and hence, give the model the ability to learn these larger displacements. We similarly hypothesized that using four frames, as opposed to two, will aid in improving performance on non-linear movements, similar to how a quadratic function can estimate a curve and a linear function cannot. We use our fully trained $\mathcal{L}_1$-model, and compare the results by measuring PSNR, IE and SSIM [5] on the two subsets of the Vimeo-90K test set containing the top 10% largest displacement and non-linear movements, called D^{90} and Q^{90} respectively. The results can be seen in Table 1.

Benchmarking General Performance In order to compare our model to others, we used several datasets covering a wide range of use cases. We first of all evaluate our model on our Large-Motion Dataset, since our main goal is to improve upon complex frame sequences. We additionally test our model on the Adobe240 and Vimeo-90K datasets, since these are regarded as the most prominent benchmarking datasets in the VI literature. We used the publicly available implementations and model weights for DAIN, both the pre-trained linear and quadratic models for QVI as publicized in [18], and the SepConv-$\mathcal{L}_1$ model. We wanted to include the current State-of-the-art, Softmax Splatting [9], but since this model is only partly released, we were not able to do this.

Method	Adobe240 [16]			Vimeo-90K [19]		
	PSNR	IE	SSIM	PSNR	IE	SSIM
DAIN [1]	31.93	7.67	0.921	34.60	5.67	0.953
QVI-lin [18]	27.07	12.05	0.861	29.68	9.22	0.904
QVI-quad [18]	31.29	7.73	0.922	33.10	6.31	0.945
SepConv-$\mathcal{L}_1$ [11]	31.62	8.16	0.917	33.74	6.15	0.945
Ours-$\mathcal{L}_1$	**32.67**	**7.21**	**0.926**	**35.25**	**5.24**	**0.954**

Table 2: Quantitative comparison of our method with other state-of-the-art methods. We see that our method is able to reach the highest performance on all metrics on the Adobe240 and Vimeo-90K datasets.

Method	LMD			Vimeo-90K [19]		
	PSNR	IE	SSIM	PSNR	IE	SSIM
Ours w/o Model Extension	28.42	14.74	0.859	34.25	5.85	0.949
Ours w/o Data Augmentations	27.75	15.14	0.852	33.98	5.95	0.945
Ours w/o Training on LMD	27.58	15.59	0.846	34.14	5.86	0.946
Ours w/o LR Scheduling	27.24	15.67	0.846	33.24	6.42	0.938
Ours	**30.21**	**14.23**	**0.861**	**35.25**	**5.24**	**0.954**

Table 3: The quantitative results of our ablation study. Best values highlighted in bold.

Ablation Study We also conducted an ablation study to individually measure the importance of some key aspects regarding model architecture and training procedure. We first of all measure the effect of the extra kernels and input frames by retraining the original SepConv-$\mathcal{L}_1$ model using our procedure, which also includes training on the Large-Motion Dataset. Since we are using the original kernel size, we load the complete set of $\mathcal{L}_1$-weights, as opposed to only loading the weights for the encoder. We also test the benefit of all our data augmentation steps and see how they affect performance. We measure the performance when we omit all data augmentations, including flipping, but leave the cropping in place. We additionally test performance when we omit training on the LMD. Lastly, we measure model performance when we do not use any learning rate scheduling, *i.e.* when we use a flat learning rate throughout training.

The results of our ablation study are shown in Table 3. We see that the removal of any of the components of our approach results in a drop in performance. As we can see, the removal of the learning rate scheduling results in a significant drop in performance. We did not expect it to have a larger impact on our performance than the removal of the data augmentation procedures.

6.2. Qualitative Analysis

We also did a thorough qualitative analysis using both test sets from the Vimeo-90K dataset and the LMD. We can

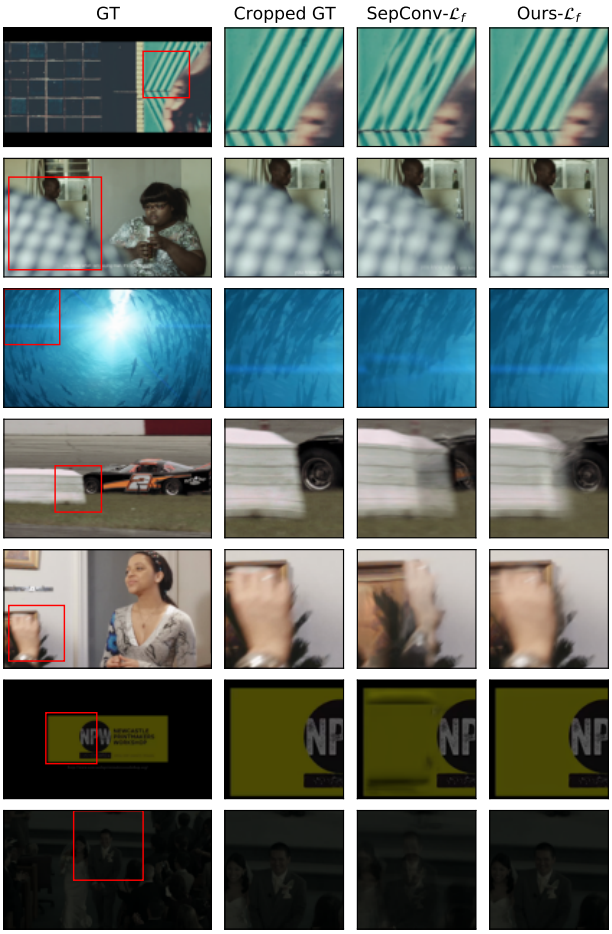

Figure 6: Visual comparison of our model on the Vimeo-90K test set. From left to right: the ground truth frame, an interesting region of the ground truth (inset in red in the first image), and the synthesized frames of SepConv-$\mathcal{L}_f$ and ours-$\mathcal{L}_f$ respectively.

confirm that we improved upon sequences containing either large displacements and/or non-linear motion in Figures 6 and 7. The frame sequences displayed contain these particular movements and we see that we manage to improve upon the results obtained by the SepConv-$\mathcal{L}_f$ model. However, we do note that while we obtain an improvement, the synthesized frames still suffer in these areas. We similarly see that our method is able to improve upon the parts of the frames containing edges, such as text and logos. The synthesized frames are clearer and contain less blur in these areas. We think this last improvement can be attributed to the learning rate scheduling procedure we employ during training to further fine-tune our weights.

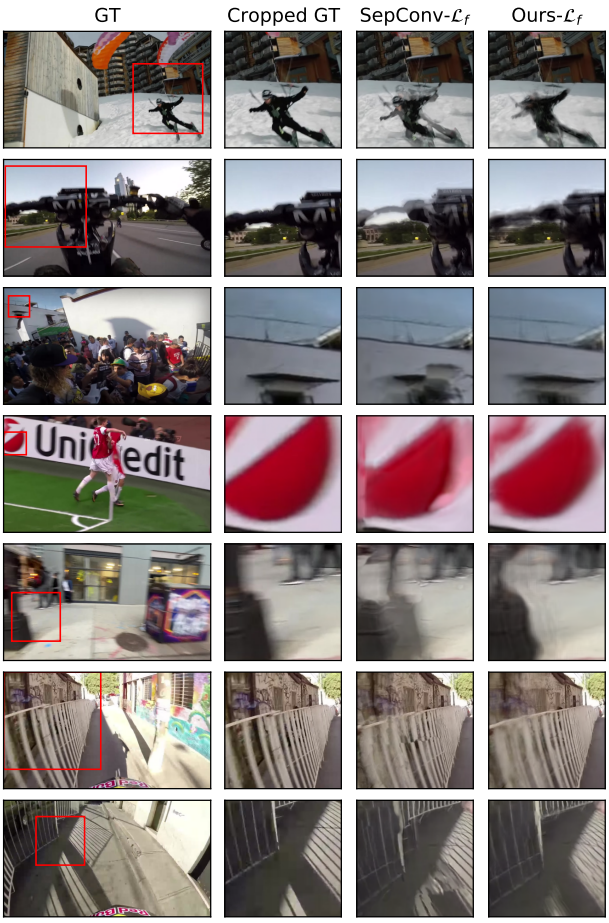

Figure 7: Visual evaluation of our model on the Large-Motion Dataset. All sequences presented here contain large movements and jittery motion.

7. Conclusion

In this paper, we introduced our approach towards solving the problems arising in large-displacement VI. Aside from showing that we are able to improve synthesis quality upon sequences that contain these problematic movements, we are also able to improve upon general performance, as shown by our quantitative benchmark on the Vimeo-90K dataset. We attribute the increased performance on large-displacement frame sequences to both the increased expresivity of the network, as well as the use of the LMD. Our abblation study shows that these steps are also necessary to achieve our result on the Vimeo-90K dataset.

References

[1] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation, 2019. 2, 4, 7

[2] James Bergstra, Daniel L K Yamins, and David D. Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. 2013. 6

[3] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011. 6

[4] Yuyu Guo, Lei Bi, Euijoon Ahn, Dagan Feng, Qian Wang, and Jinman Kim. A spatiotemporal volumetric interpolation network for 4d dynamic medical image, 2020. 1

[5] Alain Horé and Djemel Ziou. Image quality metrics: Psnr vs. ssim. pages 2366–2369, 08 2010. 7

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. 6

[7] Gucan Long, Laurent Kneip, Jose M. Alvarez, and Hongdong Li. Learning image matching by simply watching video. *CoRR*, abs/1603.06041, 2016. 2

[8] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016. 6

[9] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 7

[10] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. *CoRR*, abs/1703.07514, 2017. 2, 3, 7

[11] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 261–270, 2017. 1, 2, 3, 4, 5, 6, 7

[12] Manuel Pariente. *Ranger*, 2020. 6

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 6

[14] Jan Rühaak, Thomas Polzin, Stefan Heldmann, Ivor JA Simpson, Heinz Handels, Jan Modersitzki, and Mattias P Heinrich. Estimation of large motion in lung ct by integrating regularized keypoint correspondences into dense deformable registration. *IEEE transactions on medical imaging*, 36(8):1746–1757, 2017. 1

[15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014. 6

[16] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1279–1288, 2017. 7

[17] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. *CoRR*, abs/1709.02371, 2017. 2

[18] Xiangyu Xu, Li Siyao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. Quadratic video interpolation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 1645–1654. Curran Associates, Inc., 2019. 2, 7

[19] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T. Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, Feb 2019. 2, 3, 7