

Deep Video Frame Rate Up-conversion Network using Feature-based Progressive Residue Refinement*

Jinglei Shi, Xiaoran Jiang and Christine Guillemot

INRIA Rennes - Bretagne Atlantique, Campus Universitaire de Beaulieu, 35042 Rennes, France

Keywords: Video Frame Rate Up-conversion, Frame Interpolation, Progressive Residue Refinement, Optical Flow Estimation.

Abstract: In this paper, we propose a deep learning-based network for video frame rate up-conversion (or video frame interpolation). The proposed optical flow-based pipeline employs deep features extracted to learn residue maps for progressively refining the synthesized intermediate frame. We also propose a procedure for fine-tuning the optical flow estimation module using frame interpolation datasets, which does not require ground truth optical flows. This procedure is effective to obtain interpolation task-oriented optical flows and can be applied to other methods utilizing a deep optical flow estimation module. Experimental results demonstrate that our proposed network performs favorably against state-of-the-art methods both in terms of qualitative and quantitative measures.

1 INTRODUCTION

Video frame rate plays a critical role in video quality perception, hence in the user quality of experience in multimedia applications. Generating high frame rate videos from low frame rate versions has long been a challenging problem that has attracted a lot of attention in the computer vision community. This explains why, in recent years, a significant effort has been dedicated to the problem of video temporal interpolation for frame rate conversion. Existing video frame interpolation (VFI) approaches consist of synthesizing intermediate frames from given input frames and can be roughly classified into three categories, i.e. optical flow (Liu et al., 2017; Jiang et al., 2018; Bao et al., 2019a; Niklaus and Liu, 2018; Niklaus and Liu, 2020) or feature flow-based schemes (Gui et al., 2020), kernel-based schemes (Niklaus et al., 2017a; Niklaus et al., 2017b; Bao et al., 2019b) and phase-based ones (Meyer et al., 2015; Meyer et al., 2018).

Flow-based methods estimate optical flows between given frames, then interpolate or extrapolate target frames along the motion vectors. Liu et al. (Liu et al., 2017) proposed a Deep Voxel Flow (DVF) network that consists of a flow estimation module and a trilinear interpolation layer. The network estimates

a kind of optical flow and a temporal mask for trilinear blending the input frames. Jiang et al. (Jiang et al., 2018) propose a double U-Net pipeline where the first U-Net predicts bi-directional optical flows to be further combined to approximate the intermediate optical flows. The second U-Net refines the intermediate optical flows as well as predicts soft visibility maps in order to warp and linearly merge the input frames. Bao et al. (Bao et al., 2019a) exploit depth information in the flow projection procedure for better handling occlusions. The proposed model warps the input frames, depth maps, and contextual features based on the optical flows, and local interpolation kernels are also used for synthesizing the output frames. In (Niklaus and Liu, 2018; Niklaus and Liu, 2020), the authors employ the state-of-the-art PWC-Net (Sun et al., 2018) for optical flow estimation. Both input frames and extracted deep features are projected and merged to obtain intermediate frames. Instead of using classical optical flow estimation techniques, the authors in (Gui et al., 2020) propose a two-stage frame interpolation pipeline where feature flows estimated by a multi-flow multi-attention generator are used to warp feature maps. For methods (Bao et al., 2019a; Niklaus and Liu, 2018; Niklaus and Liu, 2020) that make use of an off-the-shelf optical flow estimation module, they both rely on a good initialization of that module whether or not end-to-end finetuning is conducted later. However, most optical flow estima-

*This work was supported by the EU H2020 Research and Innovation Programme under grant agreement No 694122 (ERC advanced grant CLIM).

tion networks are trained on synthetic datasets with ground truth optical flows, and are not fully adapted to the frame interpolation problem. In this paper, we propose a simple but effective finetuning procedure that allows us to finetune the initial optical flow estimation module with no need for ground truth optical flows.

Unlike flow-based methods, kernel-based methods (Niklaus et al., 2017a; Niklaus et al., 2017b) estimate a series of spatially-adaptive interpolation kernels, and views are then synthesized by convolving input frames with these learned kernels. In (Bao et al., 2019b), the authors integrate both optical flow and spatial compensation filters in an adaptive layer for synthesizing target frames. Kernel-based methods are computationally expensive and do not incorporate explicit mechanisms for occlusion handling. The authors in (Choi et al., 2020) employ a special feature map shuffling operation with a channel attention mechanism to replace the optical flow computation. Frames are synthesized from the re-distributed and modulated features without estimating motion information. Phase-based methods (Meyer et al., 2015; Meyer et al., 2018) are another type of motion estimation-free solutions. The motion is represented as a per pixel phase shift, hence the intermediate frame is a result of operating phase modification for each pixel. Although the authors in (Meyer et al., 2018) made progress in tackling larger motion and higher frequency content compared with (Meyer et al., 2015), they still can not reach the same level of quality as flow-based methods.

In our work, we follow the flow-based paradigm as it has yielded satisfactory results on various datasets (Soomro et al., 2012; Xue et al., 2019). Furthermore, it benefits from recent advances in optical flow estimation (Ilg et al., 2017; Hui et al., 2018; Sun et al., 2018). We use PWC-Net for optical flow estimation, that we specifically finetune for the temporal interpolation task, using the video frame interpolation dataset ‘Vimeo90K’ (Xue et al., 2019). Besides original frames, deep features extracted from these frames are also warped and used to improve the synthesis process. The authors in (Niklaus and Liu, 2018; Niklaus and Liu, 2020) also combined shallow or deep features with input frames to obtain interpolated frames, however using a complicated GridNet architecture (Fourure et al., 2017). In our scheme, feature maps at different levels are fed into ‘ConvBlocks’ to learn residue maps for a progressive refinement of the interpolated details. We have conducted comprehensive experiments on various datasets and compared with other state-of-the-art methods (Bao et al., 2019b; Gui et al., 2020; Niklaus and Liu, 2020) to show that

our design works well in the video frame rate up-conversion task.

2 METHODOLOGY

2.1 Network Overview

The video frame interpolation task aims to retrieve an intermediate frame I_t from two input frames I_0 and I_1 , where $t \in (0, 1)$ is an in-between instant. The overall architecture of our pipeline is shown in Fig. 1, where the input frames I_0 and I_1 are first fed to the flow estimator to get bidirectional optical flows $flow_{0 \rightarrow 1}$ and $flow_{1 \rightarrow 0}$, and to the encoder to obtain the feature maps $f_0^i, f_1^i (i = 1, 2, 3)$. Based on the estimated optical flows, both the input frames and their features at different resolution levels are forward projected to the target instant t . The warped images \tilde{I}_0, \tilde{I}_1 are first fed into ConvBlock1 to compute a base feature volume. Similarly, the warped feature maps at different resolution levels are fed into the ConvBlocks 2 to 4 to obtain residue feature volumes that are used to refine the base feature volume, by adding details. Both base and refined feature volumes, at the different resolution levels, are decoded by a shared decoder to generate the synthesized views \hat{I}_t^i at the target instant t with increasing image quality. The network architecture is detailed in Table 1.

2.2 Finetuned Optical Flow Estimation

As aforementioned, flow-based methods using an off-the-shelf flow estimation module always rely on a good initialization whether or not end-to-end finetuning is conducted. Available optical flow estimation networks are mostly trained on synthetic datasets with ground truth optical flows (Butler et al., 2012; Dosovitskiy et al., 2015; Ilg et al., 2018). These pre-trained networks are less adapted to real-world scene, and especially for the interpolation task. Facing this problem, we finetune the optical flow estimation network in a frame interpolation context, with the help of differentiable warping operations, in order to optimize network weights for the interpolation task. Taking PWC-Net as an example, initial bidirectional optical flows $flow_{0 \rightarrow 1}, flow_{1 \rightarrow 0}$ between I_0 and I_1 are estimated as:

$$flow_{0 \rightarrow 1}, flow_{1 \rightarrow 0} = PWC_{\theta}(I_0, I_1), \quad (1)$$

where PWC represents this motion estimation step and θ denotes the parameters in PWC-Net. The goal of finetuning is to optimize θ in order to minimize the

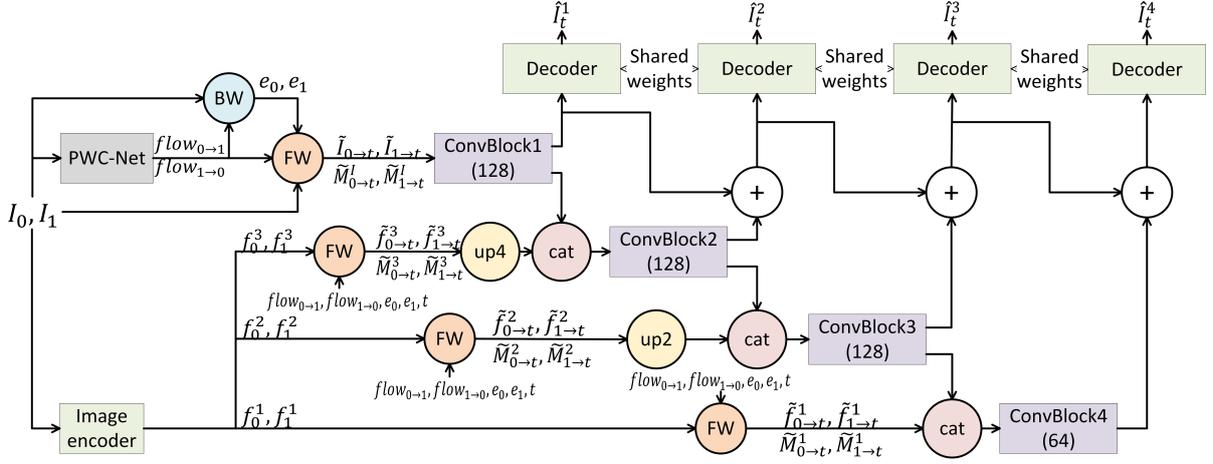


Figure 1: Overall architecture of our proposed method. Given two consecutive frames I_0 and I_1 , the flow estimator ‘PWC-Net’ and the feature extractor ‘Image encoder’ respectively predict bidirectional optical flows $flow_{0 \rightarrow 1}$, $flow_{1 \rightarrow 0}$ and extract three-scale feature maps $\{f_0^1, f_1^1, f_0^2, f_1^2, f_0^3, f_1^3\}$. Backward Warping (BW) is firstly applied by taking source frames and predicted flows to output warping errors e_0 and e_1 which are used as confidence measures. Then the Forward Warping (FW) projects the input frames and their features to the desired time instant t , where the error maps e_0 and e_1 are used to handle pixel overlaps. The feature maps in different scales bringing different levels of details are fed into ConvBlocks which generate residues that are used to progressively refine the feature volumes from the previous level (except for the lowest level). Finally, a shared decoder reconstructs a set of intermediate frames with increasing levels of details. Please refer to Table 1 for the detailed network architecture.

Table 1: Proposed network architecture. k , s and in/out represent the kernel size, the stride and the number of input/output channels, whereas ‘ \uparrow ’ and ‘ $\{$ ’ represent bilinear upsampling and concatenation.

Encoder	k	s	in/out	input
$conv1_1$	3	1	3/128	I_0 or I_1
$conv1_2$	3	1	128/128	$conv1_1$
f_0^1 or f_1^1	3	1	128/128	$conv1_2$
$conv2_1$	3	2	128/128	f_0^1 or f_1^1
$conv2_2$	3	1	128/128	$conv2_1$
f_0^2 or f_1^2	3	1	128/128	$conv2_2$
$conv3_1$	3	2	128/128	f_0^2 or f_1^2
$conv3_2$	3	1	128/128	$conv3_1$
f_0^3 or f_1^3	3	1	128/128	$conv3_2$
ConvBlock1				
$convA1$	3	1	8/128	$\{\tilde{I}_{0 \rightarrow t}, \tilde{I}_{1 \rightarrow t}, \tilde{M}_{0 \rightarrow t}^1, \tilde{M}_{1 \rightarrow t}^1\}$
$convA2$	3	1	128/128	$convA1$
D_1, R_1	3	1	128/128	$convA2$
ConvBlock2				
$convB1$	3	1	322/128	$\{\uparrow(f_0^3, f_1^3, \tilde{M}_{0 \rightarrow t}^3, \tilde{M}_{1 \rightarrow t}^3), R_1\}$
$convB2$	3	1	128/128	$convB1$
D_2, R_2	3	1	128/128	$convB2$
ConvBlock3				
$convC1$	3	1	322/128	$\{\uparrow(f_0^2, f_1^2, \tilde{M}_{0 \rightarrow t}^2, \tilde{M}_{1 \rightarrow t}^2), R_2\}$
$convC2$	3	1	128/128	$convC1$
D_3, R_3	3	1	128/128	$convC2$
ConvBlock4				
$convD1$	3	1	322/64	$\{\uparrow(f_0^1, f_1^1, \tilde{M}_{0 \rightarrow t}^1, \tilde{M}_{1 \rightarrow t}^1), R_3\}$
$convD2$	3	1	64/64	$convD1$
D_4	3	1	64/64	$convD2$
Decoder				
$convE1$	3	1	64/64	$\sum_{n=1}^4 D_n$
\tilde{I}_t^i	3	1	64/3	$convE1$

warping error:

$$\operatorname{argmin}_{\theta} |I_t - \tilde{I}_{0 \rightarrow t}|_1 + |I_t - \tilde{I}_{1 \rightarrow t}|_1, \quad (2)$$

where $\tilde{I}_{0 \rightarrow t}$ and $\tilde{I}_{1 \rightarrow t}$ are respectively warped images from given instants 0 and 1 to instant t .

There are actually two ways to obtain warped images $\tilde{I}_{0 \rightarrow t}$ and $\tilde{I}_{1 \rightarrow t}$ based on $flow_{0 \rightarrow 1}$ and $flow_{1 \rightarrow 0}$. One is to apply an ‘Image spatial transformation’ (or Backward Warping (BW)) as used in (Jaderberg et al., 2015). However, this requires interpolating $flow_{t \rightarrow 0}$ and $flow_{t \rightarrow 1}$, as in (Jiang et al., 2018), before warping the images. However, interpolating intermediate optical flows always brings errors, especially at the object boundaries. One can instead apply Forward Warping (FW) in a similar way as in (Niklaus and Liu, 2020) or (Shi et al., 2020), which can directly warp images to the time instant t without prior estimation of $flow_{t \rightarrow 0}$ and $flow_{t \rightarrow 1}$. However, the method in (Shi et al., 2020) requires a prior flow (depth maps in the case of the approach in (Shi et al., 2020)) estimation for handling overlaid pixels after warping. Overlaid pixels from foreground are given more importance than those from background when interpolating the final pixel color values.

In order to compute a measure of confidence for the warped pixels, without having to interpolate the optical flows computed between the input frames, we can, as in (Niklaus and Liu, 2020) instead calculate

brightness errors e_0 and e_1 to tackle pixel overlaps, as

$$e_0 = \sum_{RGB} |I_0 - BW(I_1, flow_{0 \rightarrow 1})|, \quad (3)$$

$$e_1 = \sum_{RGB} |I_1 - BW(I_0, flow_{1 \rightarrow 0})|. \quad (4)$$

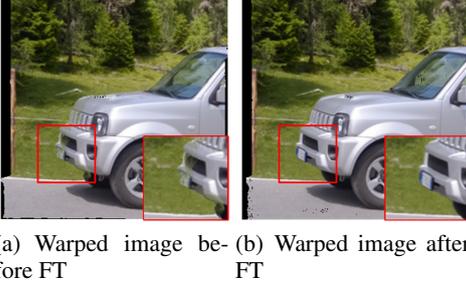


Figure 2: Visualization of $\tilde{I}_{0 \rightarrow t}$ (a) before/ (b) after finetuning (FT) flow estimator.

Warping errors e_0 and e_1 are actually effective cues for handling overlaps, since pixels coming from occluded regions have larger error values than those from visible regions, and these occluded pixels are also more likely to be overlaid after warping. So, although we use FW instead of BW to warp the input frames to the intermediate time instant t , we use BW for calculating e_0 and e_1 in Eq. 3 and Eq. 4.

With e_0 and e_1 , the authors in (Niklaus and Liu, 2020) further employ a U-Net which takes these error maps and source frames as inputs to predict importance maps for the following FW operation. However, using such a network only for predicting the pixel confidence measures implies additional parameters, and makes the fine-tuning together with PWC-Net and U-Net more complicated with risks of network collapse.

To avoid the above problems, we instead directly use the normalized error maps e'_0 and e'_1 as importance maps in the FW step to handle pixel overlaps. We adopt a scale factor α and exponential function to compute the pixel confidence measure as $exp(-\alpha * e')$. When increasing the value of α , overlaid pixels having smaller error values will be given more importance, hence will contribute more to the interpolation. A larger value of α is favorable to videos that contain large occlusions due to motion. In our experiments, we use a testset that includes different types of motion, and we experimentally found that $\alpha = 1$ can generate satisfactory results.

Besides the normalization of warping errors, we also explicitly detect disocclusion regions: after warping, the pixel positions that do not have any pixel falling in its neighbourhood, are identified as disoccluded positions. These positions are set to 0 in the

corresponding binary mask \tilde{M} , and all the others are set to 1. Based on the error maps normalization and on the detection of disocclusions, our FW can be summarized as follows:

$$\tilde{I}_{0 \rightarrow t}, \tilde{M}_{0 \rightarrow t} = FW(I_0, flow_{0 \rightarrow t}, \alpha * e'_0), \quad (5)$$

$$\tilde{I}_{1 \rightarrow t}, \tilde{M}_{1 \rightarrow t} = FW(I_1, flow_{1 \rightarrow t}, \alpha * e'_1), \quad (6)$$

where $flow_{0 \rightarrow t} = t * flow_{0 \rightarrow 1}$ and $flow_{1 \rightarrow t} = (1 - t) * flow_{1 \rightarrow 0}$.

The disocclusion masks $\tilde{M}_{0 \rightarrow t}$ and $\tilde{M}_{1 \rightarrow t}$ are crucial to our method, since they allow us to exclude unreliable pixels, which are not inpainted during warping, from the objective function Eq. 2. The new function used for finetuning the flow estimator, after taking the binary masks into account, becomes

$$argmin_{\theta} |(I_t - \tilde{I}_{0 \rightarrow t}) * \tilde{M}_{0 \rightarrow t}|_1 + |(I_t - \tilde{I}_{1 \rightarrow t}) * \tilde{M}_{1 \rightarrow t}|_1. \quad (7)$$

The disoccluded pixels are inpainted in the ConvBlocks, and the masks are indicating the disocclusion positions. Based on Eq. 7, PWC-Net can be optimized specifically for the view synthesis task. The effectiveness of our finetuning procedure is illustrated in Fig. 2, where before finetuning, we can observe severe deformations of the license plate, and blurriness due to optical flow inaccuracy. After finetuning, the deformation and blurriness issue is considerably alleviated. This is further analyzed in the ablation study section.

Apart from the improvement of the final reconstruction quality, the finetuning of PWC-Net before end-to-end training can effectively shorten global training time and prevent network collapse. Another reason for recommending this finetuning process is that when the network is not end-to-end trainable, such as in (Niklaus and Liu, 2018), having a synthesis-task optimized flow estimator definitely improves the final performance.

2.3 Progressive Residual Refinement

The next step is to retrieve the intermediate frame \hat{I}_t based on these bi-directional flows $flow_{0 \rightarrow 1}, flow_{1 \rightarrow 0}$ optimized for interpolation.

According to Eqs. 5 and 6, the original frames I_0, I_1 and extracted features f_0^i, f_1^i are first warped to instant t to obtain $\tilde{I}_{0 \rightarrow t}, \tilde{I}_{1 \rightarrow t}, \tilde{f}_{0 \rightarrow t}^i, \tilde{f}_{1 \rightarrow t}^i$ and their corresponding binary masks $\tilde{M}_{0 \rightarrow t}^i, \tilde{M}_{1 \rightarrow t}^i$. ConvBlock1 first takes warped images and their masks and outputs a feature volume containing a base representation of scene information, which we then refine by adding residue feature volumes based on warped features $f_{0 \rightarrow t}^i, \tilde{f}_{1 \rightarrow t}^i$ at different levels obtained with kernels having different receptive fields. These features aim at capturing subtle textures in the scene, to

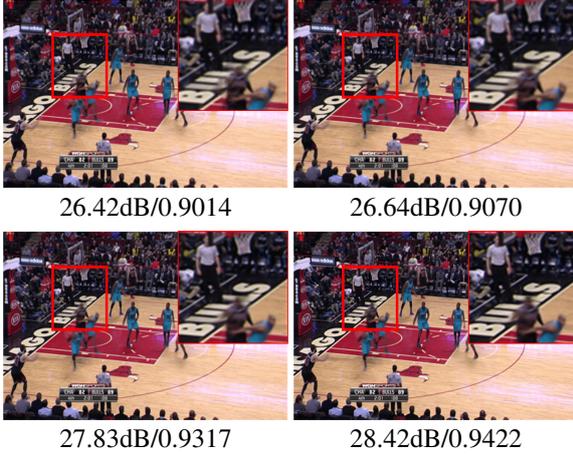


Figure 3: Visualization of progressively refined details from \hat{I}_t^1 to \hat{I}_t^4 (from top to bottom, from left to right).

then refine details of the base feature volume. The merging of images resulting from both pixel-based and feature-based warping has been shown powerful in (Shi et al., 2020) in the context of light field view synthesis. The features extracted at three different scales are used, via ConvBlock2-4, to learn residues to progressively refine the base feature map volume. ConvBlock1-4 are three-layer convolutional blocks. The last layer of ConvBlock1-3 is split into two branches, one branch keeping a subset of features that refines the previous level, while the second subset of features (corresponding to higher frequency details) is used at the next refinement level (see Fig 1(c)). Finally, a shared decoder is employed to decode both the base volume and volumes progressively refined at different levels to obtain synthesized views $\hat{I}_t^i (i = 1, 2, 3, 4)$. Fig. 3 shows this progressive refinement from \hat{I}_t^1 to \hat{I}_t^4 . From \hat{I}_t^1 to \hat{I}_t^4 , more and more details are added and artefacts are gradually corrected. We use \hat{I}_t^4 as our final interpolated frame as it contains the most details. The whole pipeline is trained by minimizing the following reconstruction error:

$$L_{rec} = \sum_{i=1}^4 \lambda_i Lap(I_t - \hat{I}_t^i), \quad (8)$$

where Lap is the Laplacian loss (Bojanowski et al., 2017) with three levels, and λ_i are hyperparameters for controlling the reconstruction quality at each level. More precisely, we set λ_i to non-zero values at the beginning of the training, to enforce a target frame reconstruction at each network level. Once each network level is well initialized, we set λ_4 to non-zero and the other λ values to zero in order to only focus on the last reconstruction step.

3 TRAINING DETAILS

We use tensorflow to implement our method and Adam to optimize the model with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The overall training schedule of our method is divided in two steps: first finetuning PWC-Net and then training the whole network.

The finetuning of PWC-Net aims at optimizing its weights for the synthesis task. In this step, we set the batch size to 20, the patch size to 160×160 and the learning rate to 10^{-4} to finetune PWC-Net for 20 epochs. We found that 20 epochs of finetuning can already correct most of deformations and blurriness. More epochs will prolong the overall schedule while bringing limited improvement. We found that this finetuning procedure is helpful to the global schedule, since the following training procedure will be much longer, and may suffer from performance oscillation or even collapse without a finetuned flow estimator.

The training of the whole network is made in two steps. We first fix the weights in PWC-Net and update the other variables, with a batch size of 20, a patch size of 160×160 , a learning rate of 10^{-4} and $\lambda_1 \dots \lambda_4 = \{0.1, 0.2, 0.4, 0.8\}$ for 50 epochs. This step guides the ConvBlock1-4 layers to learn residues for correcting details in a coarse to fine manner. In the second step, we end-to-end train the network and update all variables, with a batch size of 16, a patch size of 160×160 , a learning rate of 10^{-5} and $\lambda_4 = 1$ for 40 epochs. $\lambda_1, \lambda_2, \lambda_3$ are set to 0 to make the optimization focus on the final synthesized view. The training of our pipeline has been conducted on a Nvidia Tesla V100 GPU card with 32GB GRAM, and took about 6 days to converge.

4 EXPERIMENTAL RESULTS

We have carried out experiments on several datasets and measured the PSNR and SSIM of the interpolated frames in comparison with state-of-the-art methods.

4.1 Datasets

We used the following datasets:

- **Training Dataset.** Both finetuning of PWC-Net and training of the network have been conducted using the Vimeo90K training set (Xue et al., 2019), which contains 51,312 frame triplets with resolution 256×448 . We use the first and the last frames as inputs and the intermediate frame as ground truth.
- **Test Datasets.** We have used two test sets to assess the quality of the synthesized frames: 1) the

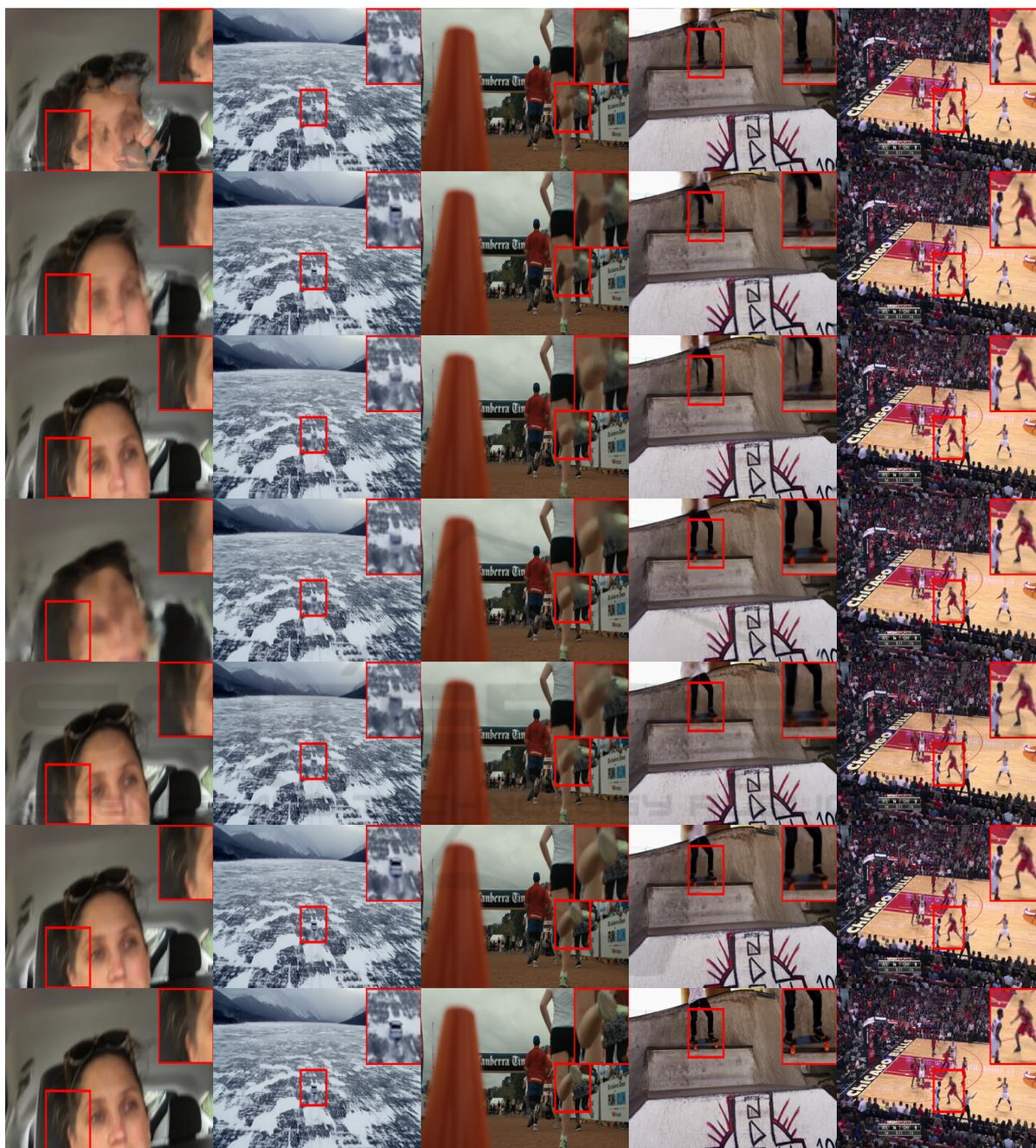


Figure 4: Visual comparison of synthesized frames for different methods. From top to bottom, they are frames interpolated using SuperSloMo (Jiang et al., 2018), SepConv (Niklaus et al., 2017b), MEMC (Bao et al., 2019b), FeFlow (Gui et al., 2020), SMSP (Niklaus and Liu, 2020), Our method. The last row shows ground truth frames.

Vimeo90K testset which has 3,782 frame triplets with resolution 256×448 ; 2) the Adobe240fps dataset is a dataset captured by a handheld camera. We have extracted 777 frame triplets and interpolated the intermediate frame during testing.

4.2 Experimental Set Up

We compare the proposed video frame interpolation method with the most representative and recent methods: SuperSloMo (Jiang et al., 2018), SepConv (Niklaus et al., 2017b), MEMC (Bao et al., 2019b), FeFlow (Gui et al., 2020) and SMSP (Niklaus and

Table 2: Quantitative results (PSNR&SSIM) for the interpolated frames (averaged over 3782 and 777 frames).

Methods	Vimeo90K		Adobe240fps	
	PSNR	SSIM	PSNR	SSIM
<i>SuperSloMo</i> (Jiang et al., 2018)	30.92	0.9320	28.44	0.8966
<i>SepConv</i> (Niklaus et al., 2017b)	33.80	0.9555	31.16	0.9225
<i>MEMC</i> (Bao et al., 2019b)	34.43	0.9625	31.54	0.9269
<i>FeFlow</i> (Gui et al., 2020)	35.09	0.9629	31.50	0.9248
<i>SMSP</i> (Niklaus and Liu, 2020)	<u>35.49</u>	<u>0.9671</u>	<u>31.62</u>	<u>0.9269</u>
<i>Ours</i>	35.86	0.9689	31.80	0.9286

Liu, 2020). Among them, SuperSloMo infers the optical flow in the target instant to backward project the input frames and employs soft visibility maps to handle occlusions. SepConv formulates the frame interpolation problem as a local separable convolution using 1D kernels. Target frames are synthesized without involving any optical flow. MEMC predicts both optical flows and convolution kernels respectively for global motion estimation and local motion compensation in its synthesis process. FeFlow pioneers the flows of feature maps instead of images to retrieve an intermediate frame. SMSP builds a two-stage synthesis pipeline by using an off-the-shelf PWC-Net for flow prediction, and a GridNet (Fourure et al., 2017) for frame synthesis, which is in the same vein as ours.

For a fair comparison, we use the official authors' implementations for the methods SepConv (Niklaus et al., 2017b), MEMC (Bao et al., 2019b) and FeFlow (Gui et al., 2020). We use a third-party code and a pre-trained model for the SuperSloMo (Jiang et al., 2018) method. For the SMSP method (Niklaus and Liu, 2020), the authors only provide the implementation of their FW operation. The rest of the network implementation and the trained models are not available. Therefore, we re-implemented and trained their network architecture by following the instructions in the paper. Since MEMC, FeFlow and SMSP are already trained on Vimeo90K dataset and SuperSloMo is trained on Adobe240fps datasets, we use the default parameter settings recommended by the authors in our experiments, e.g. we use kernel size 4×4 in adaptive warping layer of MEMC, 16 groups of attention maps in FeFlow etc.

4.3 Interpolation Results

The performances of all tested methods are shown in Table 2. We can observe that our method outperforms the best referenced methods by a margin of about 0.2-0.3dB. This gain of PSNR is not trivial, as we can notice that the difference in terms of PSNR between the reference methods is sometimes less than 0.1dB.

Besides the quantitative evaluation in terms of PSNR and SSIM, Fig 4 shows synthesized views us-

ing different methods. We zoomed regions containing subtle details in each image for comparison. We can notice that our method better reconstructs details when compared with other methods. For details like the hair in the first scene and the car in the second scene, where other methods suffer from a deformation or blurriness, our pipeline yields more plausible results.

Another advantage of our method is that it can up-convert videos to an arbitrary frame rate, since the target instant t can be any value between 0 and 1. Methods such as SepConv and FeFlow can only interpolate frames at $t = 0.5$, which means that, to interpolate by a factor of $N = 2^k$ ($k > 1$), they must recursively perform k interpolations. Interpolation errors will gradually augment during this recursive synthesis process. We have 8X up-converted video frame rate using different methods, and the corresponding interpolated videos can be found in our project homepage: <http://clim.inria.fr/research/VISAPP2022/index.html>, where high frame rate videos obtained using our pipeline have less artifacts and deformations than others.

Please note that, like most of the frame synthesis methods, our method is based on linear-motion assumption, all motions are supposed to have uniform velocities during frame warping process. Inferring both velocity and acceleration of motion using only two frames is a very ill-posed problem, it is more investigated in a multi-frame (Bao et al., 2018; Reda et al., 2019; Xu et al., 2019) context.

4.4 Ablation Study

4.4.1 Forward Warping Method

Although both our method and SMSP are built on PWC-Net, the rest of the network has a different design. In SMSP, the authors employ a neural network that takes brightness errors e_0, e_1 in Eq.3 and Eq.4 to learn maps of importance for handling pixel overlaps, this network increases the total parameter number but brings little improvement. While we normalize two brightness errors as maps of importance, which is

Table 3: Quantitative evaluation of different finetuning strategies on Vimeo90K testset. The obtained PSNR values are averaged over 3782 frames.

Types	wo FT	FT(SMSP FW)	FT(our FW)
PSNR	31.26	33.38	33.81

likewise effective in handling pixel overlaps but without an increasing parameter number.

Fig. 5 shows the warped images obtained when using the FW step proposed in SMSP (Niklaus and Liu, 2020). Although the quality of the two warping methods looks similar, the proposed method is very simple, as it does not use extra network as in (Niklaus and Liu, 2020). We have also quantitatively evaluated the performance when using different FW strategies. Table 3 shows averaged PSNR over 3782 frames. More specially, we use optical flows to warp the first and the third frames to the intermediate instant, and compare it with ground truth frame. In this process, all disoccluded pixels are excluded. The employed optical flows are estimated without finetuning or with finetuning, using FW proposed in (Niklaus and Liu, 2020), or finetuned using our proposed FW. Our proposed simple FW step makes the end-to-end fine tuning of all the network components of the architecture easier, leading to a better quality for the final interpolated frames.



(a) Warped image using FW proposed in (Niklaus and Liu, 2020) (b) Warped image using our proposed FW

Figure 5: Visualization of $\tilde{I}_{0 \rightarrow t}$ using (a) FW proposed in (Niklaus and Liu, 2020) and (b) Our FW.

4.4.2 Refinement Structure and Parameters

Compared with the complex GridNet model of (Fourure et al., 2017), our progressive refinement block is simple but efficient, it focuses more on the details of the retrieved frames. Fine details are better reconstructed with the help of learned residues. Using fewer parameters (1.9M vs 3.0M), our method finally achieves better performance (in Table 2) and visual quality (in Fig.4) than SMSP.

To investigate the impact of using multi-levels refinement, we carry out experiments either using only the base layer (with the warped input frames only),

Table 4: Quantitative evaluation of different levels of refinement. The obtained PSNR values are averaged over 3782 frames of the Vimeo90K testset.

Types	base	1-layer	2-layer	full model
PSNR	35.05	35.19	35.40	35.86

or using one, two, or three levels of refinement of the warped images features. Table 4 shows averaged PSNR values when adopting different levels of refinement. The full model using three layers of refinement gives the best performance.

5 CONCLUSION

In this work, we proposed a flow-based network architecture that uses deep features to progressively refine a temporally interpolated frame in a context of video frame rate up-conversion. We also proposed a novel finetuning procedure that optimizes flow estimation networks for the interpolation task without using any ground truth optical flows. A comprehensive qualitative and quantitative assessment on different video frame interpolation datasets shows that our method can generate high quality interpolated frames with realistic details.

REFERENCES

- Bao, W., Lai, W., Ma, C., Yang, M., et al. (2019a). Depth-aware video frame interpolation. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3703–3712.
- Bao, W., Lai, W., Zhang, X., Yang, M., et al. (2019b). Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*.
- Bao, W., Zhang, X., Chen, L., Ding, L., and Gao, Z. (2018). High-order model and dynamic filtering for frame rate up-conversion. *IEEE Trans. Image Proc. (TIP)*, 27(8):3813–3826.
- Bojanowski, P., Joulin, A., Lopez-Paz, D., and Szlam, A. (2017). Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*.
- Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In *Eu. Conf. on Computer Vision (ECCV)*, pages 611–625.
- Choi, M., Kim, H., Han, B., Xu, N., and Lee, K. (2020). Channel attention is all you need for video frame interpolation. In *AAAI Conference on Artificial Intelligence*, pages 10663–10671.
- Dosovitskiy, A., Fischer, P., Ilg, E., et al. (2015). FlowNet: Learning optical flow with convolutional networks. In *IEEE Int. Conf. on Computer Vision (ICCV)*.

- Fourure, D., Emonet, R., Fromont, E., et al. (2017). Residual conv-deconv grid network for semantic segmentation. In *British Machine Vision Conf. (BMVC)*.
- Gui, S., Wang, C., Chen, Q., and Tao, D. (2020). Featureflow: Robust video interpolation via structure-to-texture generation. In *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 14004–14013.
- Hui, T., Tang, X., and Loy, C. C. (2018). Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 8981–8989.
- Ilg, E., Mayer, N., Brox, T., et al. (2017). Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2462–2470.
- Ilg, E., Saikia, T., Keuper, M., and Brox, T. (2018). Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *Eu. Conf. on Computer Vision (ECCV)*.
- Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2017–2025.
- Jiang, H., Sun, D., Jampani, V., Kautz, J., et al. (2018). Super slo-mo: High quality estimation of multiple intermediate frames for video interpolation. In *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9000–9008.
- Liu, Z., A, R. Y., Tang, X., Liu, Y., and Agarwala, A. (2017). Video frame synthesis using deep voxel flow. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 4463–4471.
- Meyer, S., Djelouah, A., McWilliams, B., Schroers, C., et al. (2018). Phasenet for video frame interpolation. In *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 498–507.
- Meyer, S., Wang, O., Zimmer, H., Sorkine-Hornung, A., et al. (2015). Phase-based frame interpolation for video. In *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1410–1418.
- Niklaus, S. and Liu, F. (2018). Context-aware synthesis for video frame interpolation. In *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1701–1710.
- Niklaus, S. and Liu, F. (2020). Softmax splatting for video frame interpolation. In *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5437–5446.
- Niklaus, S., Mai, L., and Liu, F. (2017a). Video frame interpolation via adaptive convolution. In *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 670–679.
- Niklaus, S., Mai, L., and Liu, F. (2017b). Video frame interpolation via adaptive separable convolution. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 261–270.
- Reda, F., Sun, D., Dundar, A., Shoyebi, M., Liu, G., Shih, K., Tao, A., Kautz, J., and Catanzaro, B. (2019). Unsupervised video interpolation using cycle consistency. In *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 892–900.
- Shi, J., Jiang, X., and Guillemot, C. (2020). Learning fused pixel and feature-based view reconstructions for light fields. In *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2555–2564.
- Soomro, K., Zamir, A. R., and Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.
- Sun, D., Yang, X., Liu, M., and Kautz, J. (2018). Pwcnet: Cnns for optical flow using pyramid, warping, and cost volume. In *IEEE. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943.
- Xu, X., Siyao, L., Sun, W., Yin, Q., and Yang, M. (2019). Quadratic video interpolation. *Advances in Neural Information Processing Systems (NIPS)*.
- Xue, J., Chen, B., T, W., et al. (2019). Video enhancement with task-oriented flow. *Int. J. Computer Vision (IJCV)*, 127(8):1106–1125.