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

Jinglei Shi, Xiaoran Jiang, Christine Guillemot

To cite this version:

HAL Id: hal-03432380
https://hal.science/hal-03432380
Submitted on 17 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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
name.surname@inria.fr

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 estimation 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 synthetized 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 synthetized 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 $\tilde{I}_0^1$, $\tilde{I}_1^1$ 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 $I'_t$ 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 fine-tuning 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:

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

where $PWC$ represents this motion estimation step and $\theta$ denotes the parameters in PWC-Net. The goal
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 \( \text{flow}_{0 \rightarrow 1} \), \( \text{flow}_{1 \rightarrow 0} \) and extract three-scale feature maps \( \{ f_{1,0}, f_{1,1}, f_{2,0}, f_{2,1}, f_{3,0}, f_{3,1} \} \). 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 decoded reconstruct a set of intermediate frames with increasing levels of details. Please refer to Table 1 for the detailed network architecture.

of finetuning is to optimize \( \theta \) in order to minimize the warping error:

\[
\arg\min_{\theta} |I_t - \tilde{I}_{0 \rightarrow t}|_1 + |I_t - \tilde{I}_{1 \rightarrow t}|_1, \tag{2}
\]

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 \( \text{flow}_{0 \rightarrow 1} \) and \( \text{flow}_{1 \rightarrow 0} \). One is to apply an ‘Image spatial transformation’ (or Backward Warping (BW)) as used in [Jaderberg et al., 2015]. However, this requires interpolating \( \text{flow}_{0 \rightarrow 1} \) and \( \text{flow}_{1 \rightarrow 0} \), 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 \( \text{flow}_{0 \rightarrow 1} \) and \( \text{flow}_{1 \rightarrow 0} \). 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

\[
e_0 = \sum_{\text{RGB}} |I_0 - \text{BW}(I_1, \text{flow}_{0 \rightarrow 1})|, \tag{3}
\]

\[
e_1 = \sum_{\text{RGB}} |I_1 - \text{BW}(I_0, \text{flow}_{1 \rightarrow 0})|. \tag{4}
\]
falling in its neighbourhood, are identified as disocclusions, the pixel positions that do not have any pixel overlap. The disocclusion masks \( M_{0 \rightarrow t} \) and \( M_{1 \rightarrow t} \) 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:

\[
\arg\min_{\beta}[\|I_t - \tilde{I}_{0 \rightarrow t}\|^2 + \|I_t - \tilde{I}_{1 \rightarrow t}\|^2 + \|	ilde{M}_{0 \rightarrow t}\| + \|	ilde{M}_{1 \rightarrow t}\|].
\]

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 \( \tilde{I}_t \) based on these bi-directional flows \( \text{flow}_{0 \rightarrow 1}, \text{flow}_{1 \rightarrow 0} \) optimized for interpolation.

According to Eqs. 5 and 6, the original frames \( I_0, I_1 \) and extracted features \( f^0, f^1 \) are first warped to instant \( t \) to obtain \( \tilde{I}_{0 \rightarrow t}, \tilde{I}_{1 \rightarrow t}, \tilde{f}^0_{0 \rightarrow t}, \tilde{f}^1_{1 \rightarrow t} \) and their corresponding binary masks \( \tilde{M}^0_{0 \rightarrow t}, \tilde{M}^1_{1 \rightarrow t} \). 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 \( \tilde{f}^0_{0 \rightarrow t}, \tilde{f}^1_{1 \rightarrow t} \) at different levels obtained with

<table>
<thead>
<tr>
<th>Encoder</th>
<th>k</th>
<th>s</th>
<th>in/out</th>
<th>input</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv_1</td>
<td>3</td>
<td>1</td>
<td>4/128</td>
<td>( \tilde{I}_0, I_1 )</td>
</tr>
<tr>
<td>conv_2</td>
<td>3</td>
<td>1</td>
<td>128/128</td>
<td>( \tilde{I}_0, I_1 )</td>
</tr>
<tr>
<td>( f^0 ) or ( f^1 )</td>
<td>3</td>
<td>1</td>
<td>128/128</td>
<td>( \text{conv}_2 )</td>
</tr>
<tr>
<td>conv_2</td>
<td>3</td>
<td>2</td>
<td>128/128</td>
<td>( f^0 ) or ( f^1 )</td>
</tr>
<tr>
<td>( f^0 ) or ( f^1 )</td>
<td>3</td>
<td>1</td>
<td>128/128</td>
<td>( \text{conv}_2 )</td>
</tr>
<tr>
<td>conv_2</td>
<td>3</td>
<td>2</td>
<td>128/128</td>
<td>( f^0 ) or ( f^1 )</td>
</tr>
<tr>
<td>( f^0 ) or ( f^1 )</td>
<td>3</td>
<td>1</td>
<td>128/128</td>
<td>( \text{conv}_2 )</td>
</tr>
<tr>
<td>( f^0 ) or ( f^1 )</td>
<td>3</td>
<td>2</td>
<td>128/128</td>
<td>( f^0 ) or ( f^1 )</td>
</tr>
<tr>
<td>( f^0 ) or ( f^1 )</td>
<td>3</td>
<td>1</td>
<td>128/128</td>
<td>( \text{conv}_2 )</td>
</tr>
<tr>
<td>( f^0 ) or ( f^1 )</td>
<td>3</td>
<td>1</td>
<td>128/128</td>
<td>( \text{conv}_2 )</td>
</tr>
<tr>
<td>( f^0 ) or ( f^1 )</td>
<td>3</td>
<td>2</td>
<td>128/128</td>
<td>( f^0 ) or ( f^1 )</td>
</tr>
<tr>
<td>( f^0 ) or ( f^1 )</td>
<td>3</td>
<td>1</td>
<td>128/128</td>
<td>( \text{conv}_2 )</td>
</tr>
<tr>
<td>( f^0 ) or ( f^1 )</td>
<td>3</td>
<td>1</td>
<td>128/128</td>
<td>( \text{conv}_2 )</td>
</tr>
<tr>
<td>( f^0 ) or ( f^1 )</td>
<td>3</td>
<td>2</td>
<td>128/128</td>
<td>( f^0 ) or ( f^1 )</td>
</tr>
</tbody>
</table>

Table 1: Proposed network architecture. \( k \), \( s \) and \( \text{in/out} \) represent the kernel size, the stride and the number of input/output channels, whereas ‘+’ and ‘{ }’ represent bilinear upsampling and concatenation.
is trained by minimizing the following reconstruction as it contains the most details. The whole pipeline

\[ \lambda \]

More precisely, we set

\[ \lambda \]

for controlling the reconstruction quality at each level. \[\lambda \] with three levels, and

\[ \text{Lap} \] and more details are added and artefacts are gradually

sive refinement from \( \hat{I}_t \). Finally, a shared decoder is employed to decode both the base volume and volumes progressively refined at different levels to obtain synthesized views \( \hat{I}_t \). From \( \hat{I}_1 \) to \( \hat{I}_4 \), more and more details are added and artefacts are gradually corrected. We use \( \hat{I}_4 \) as our final interpolated frame as it contains the most details. The whole pipeline is trained by minimizing the following reconstruction error:

\[ L_{rec} = \sum_{i=1}^{4} \lambda_i \text{Lap}(I_t - \hat{I}_i^t), \]  

(8)

where \( \text{Lap} \) is the Laplacian loss \cite{Bojanowski et al., 2017} with three levels, and \( \lambda_i \) are hyperparameters for controlling the reconstruction quality at each level. More precisely, we set \( \lambda_1 \) 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 \( \lambda_4 \) to non-zero and the other \( \lambda \) 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 \( 160 \times 160 \) 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 \( 160 \times 160 \), a learning rate of \( 10^{-4} \) and \( \lambda_1 \ldots \lambda_4 = \{0.1, 0.2, 0.4, 0.8\} \) 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 \( 160 \times 160 \), 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 \cite{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.
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.

**Test datasets.** We have used two test sets to assess the quality of the synthesized frames: 1) the 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
Table 2: Quantitative results (PSNR&SSIM) for the interpolated frames (averaged over 3782 and 777 frames).

<table>
<thead>
<tr>
<th>Methods</th>
<th>Vimeo90K</th>
<th>Adobe240fps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR</td>
<td>SSIM</td>
</tr>
<tr>
<td>SuperSloMo [Jiang et al., 2018]</td>
<td>30.92</td>
<td>0.9320</td>
</tr>
<tr>
<td>SepConv [Niklaus et al., 2017b]</td>
<td>33.80</td>
<td>0.9555</td>
</tr>
<tr>
<td>MEMC [Bao et al., 2019b]</td>
<td>34.43</td>
<td>0.9625</td>
</tr>
<tr>
<td>FeFlow [Gui et al., 2020]</td>
<td>35.09</td>
<td>0.9629</td>
</tr>
<tr>
<td>SMSP [Niklaus and Liu, 2020]</td>
<td>35.49</td>
<td>0.9671</td>
</tr>
<tr>
<td>Ours</td>
<td>35.86</td>
<td>0.9689</td>
</tr>
</tbody>
</table>

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 using 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.

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

<table>
<thead>
<tr>
<th>Types</th>
<th>wo FT</th>
<th>FT(SMSP FW)</th>
<th>FT(our FW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>31.26</td>
<td>33.38</td>
<td>33.81</td>
</tr>
</tbody>
</table>

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 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.

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.

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

<table>
<thead>
<tr>
<th>Types</th>
<th>base</th>
<th>1-layer</th>
<th>2-layer</th>
<th>full model</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>35.05</td>
<td>35.19</td>
<td>35.40</td>
<td>35.86</td>
</tr>
</tbody>
</table>

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), 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


