Learning to Extract Flawless Slow Motion from Blurry Videos

Meiguang Jin
AutoNavi
meiguang.jmg@autonavi.com

Zhe Hu
Hikvision Research
zhe.hu@hikvision.com

Paolo Favaro
University of Bern
paolo.favaro@inf.unibe.ch

Abstract

In this paper, we introduce the task of generating a sharp slow-motion video given a low frame rate blurry video. We propose a data-driven approach, where the training data is captured with a high frame rate camera and blurry images are simulated through an averaging process. While it is possible to train a neural network to recover the sharp frames from their average, there is no guarantee of the temporal smoothness for the formed video, as the frames are estimated independently. To address the temporal smoothness requirement we propose a system with two networks: One, DeblurNet, to predict sharp keyframes and the second, InterpNet, to predict intermediate frames between the generated keyframes. A smooth transition is ensured by interpolating between consecutive keyframes using InterpNet. Moreover, the proposed scheme enables further increase in frame rate without retraining the network, by applying InterpNet recursively between pairs of sharp frames. We evaluate the proposed method on several datasets, including a novel dataset captured with a Sony RX V camera. We also demonstrate its performance of increasing the frame rate up to 20 times on real blurry videos.

1. Introduction

The understanding of fast moving objects is a challenging task that the average human eye can tackle only through the aid of specialized hardware. Fortunately, commercial cameras that can capture events in extreme slow-motion have been made available by recent advances in the technology. At one end of the camera spectrum are very expensive and high-quality devices, used mostly by movie makers. At the other end of the spectrum are cheaper commercial products that currently offer up to 240 frames per second (FPS). The latter cameras tend to suffer from low signal-to-noise ratio. This problem, however, is a general challenge, since an increase of the frame rate causes a reduction of the exposure and, consequently, of the amount of light impinging on the sensor, which leads to poor signal to noise ratio. Another challenge of these cameras is that they require enormous transfer bandwidth and storage space. Moreover, with an increasing frame rate, frames share a lot of content, which makes the whole capture process resource-wasteful.

A cheaper and more resource-efficient alternative to the high frame rate and high noise-sensitivity hardware solution is to use low frame rate cameras with long exposures and increase their frame rate computationally. This has been addressed by developing algorithms to interpolate subsequent frames so that objects in the scene move naturally [17]. However, using long exposures to image moving objects may result in motion blur. Thus, simply interpolating frames of a low frame rate camera may result in unrealistic high frame rate blurry videos. To synthesize the frames captured with a high frame rate camera and its small exposure time, it is also necessary to perform motion deblurring. A direct solution is to combine the existing state of the art methods for video deblurring (*e.g.*, [24]), which yield sharp frames corresponding to each input blurry frame, and those for video interpolation (*e.g.*, [6]), which generate intermediate frames between the deblurred frames, sequentially. However, a naïve combination of these methods is suboptimal, because the deblurring process eliminates useful temporal information. In fact, blurry frames contain information about the intermediate sharp frames (as their average), and once deblurred, the recovered frame will contain information only about one of the intermediate frames.

To address these issues, we propose the system illustrated in Fig. 1. The method is built as a combination of deblurring and interpolation, by introducing two neural networks: DeblurNet and InterpNet. The first network takes as input four blurry frames and outputs five sharp keyframes, which lie between the two middle input frames (highlighted in blue in Fig. 1(a)). The second network takes as input two subsequent keyframes and two blurry inputs, and generates a sharp frame between the two keyframes. InterpNet brings three benefits: 1) it generates realistic intermediate frames by exploiting the motion information stored in the input blurry frames; 2) it ensures a smooth temporal transition between output frames (see "frame across inputs" in Fig. 1(b)) by using keyframes from subsequent blurry frames quadruplets; 3) it allows to interpolate an arbitrary

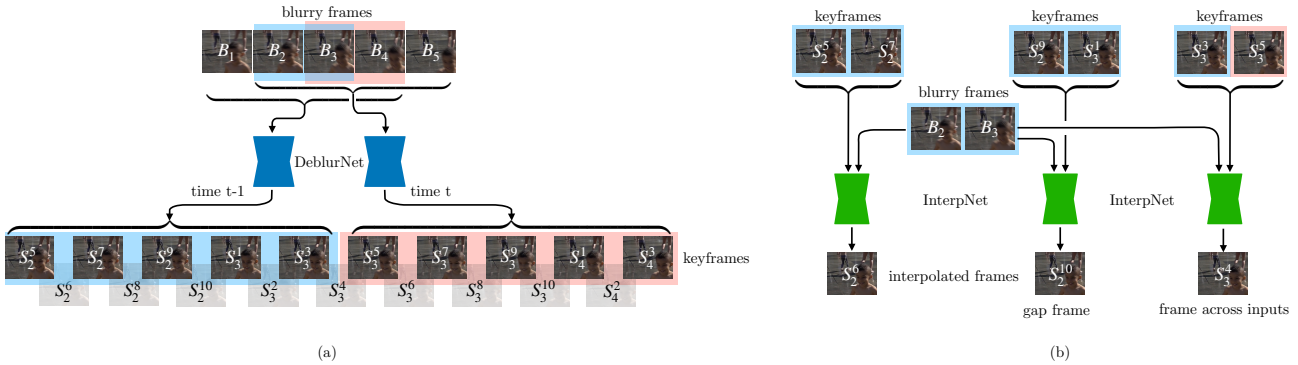


Figure 1: (a) DeblurNet outputs keyframes of the video. (b) InterpNet outputs the intermediate frames between the keyframes.

output frame rate, through its recursive application to pairs of output frames. With reference to Fig. 1, InterpNet could also be applied to pairs of subsequent output frames (*e.g.*, between the keyframe S_2^5 and the interpolated frame S_2^6) to generate a new intermediate frame. In the experiments, we show each of these benefits and, in particular, the ability of our method to achieve a 20-fold increase in the frame rate.

To guarantee the success of training, as was done in recent works on video deblurring, we build a synthetic dataset of blurry/sharp images. Sharp images are obtained by capturing videos with a high FPS camera and the corresponding blurry frames are obtained by averaging those images and adding noise. Instead of using existing datasets, which are captured using smartphone cameras and GoPros [7, 16, 19, 24, 25], we collect a new dataset consisting of 1080P videos at 250 FPS by using a Sony RX V camera. The images captured by smartphone cameras and GoPros tend to have low signal-to-noise ratio. In contrast, our dataset provides better image quality due to the large pixel size of the Sony camera sensor. We will release this dataset to the community, to contribute and foster future research on video deblurring and interpolation algorithms.

Our main contributions are summarized as follows:

1. To the best of our knowledge, this is the first work to generate sharp slow-motion videos from blurry videos;

2. We provide a novel high-quality video dataset, which can be used for video deblurring, video frame interpolation and the joint problem proposed and addressed in this paper;

3. We introduce novel technical components that: 1) ensure the temporal smoothness of the output sequence; 2) exploit motion embedded in each motion-blurred frame; 3) enable the generation of videos with arbitrary frame rates; 4) yield higher quality slow-motion videos than in prior work.

2. Related Work

Motion Deblurring. Motion deblurring is an extremely challenging problem due to its ill-posed nature. Classical approaches formulate image deblurring problems with a space-invariant blur model and optimize it by introducing image priors/regularizations that characterize natural image statistics to address the ill-posedness [1, 3, 8, 20, 22, 27, 28]. However, real-world blurry images typically contain space-varying blur [26], due to a depth variation in the scene and the non-translational motion of the camera and objects. Some works address the non-uniform deblurring problem by simultaneously recovering blur kernels and scene depth [5, 21] or via scene segmentation [9]. Recently, convolutional neural networks (CNNs) have been applied to bypass the computationally expensive kernel estimation step, and achieve impressive results [4, 7, 16, 18, 19, 25]. These methods often decrease contrast in areas with low contrast and therefore generate results with cartoon-like effects. To recover the missing high-frequency details during the blur process, the method [13] adopts a Wasserstein generative adversarial network (GAN) to render realistic deblurred images.

An extension of the above work is video deblurring, where the sharpening needs to be applied to a sequence of blurry frames about the same scene. The pioneering work of Cho *et al*. [2] explores the similarity between frames of the same video and exploits sharp patches from neighboring frames. In [11], an integrated model is proposed to jointly predict the defocus blur, optical flow and latent frames. Recently, with the advance of deep learning, great progress has been made in video deblurring. [24] deploys a U-net structure that takes five consecutive blurry frames and returns the deblurred third frame. By exploiting a recurrent architecture, [10] achieves real time performance.

Frame Interpolation. Frame interpolation is commonly used for frame rate conversion, image morphing, and motion-based video compression. Conventional approaches address the problem by estimating optical flow between in-

put frames and interpolating the frames of interest via some given image formation model [15, 31]. These optical-flow-based approaches are able to render intermediate frames at any positions between input frames. However, the quality of the interpolated images heavily depends on the accuracy of the optical flow estimate. Recent methods adopt data-driven approaches to estimate the dense correspondence and render the interest frames in end-to-end CNNs [14, 30]. Unlike flow-based methods, [17] treats each pixel of the interpolated frame as a local convolution between two corresponding patches of the input frames, and learns a CNN to estimate space-varying and separable convolutional kernels for each pixel. To enable multi-frame interpolation, a flow-based CNN is proposed with a special handling of occlusions [6]. Although these CNN-based interpolation methods achieve impressive results on sharp input frames, they are not able to produce accurate results when applied to input frames degraded by blur.

To the best of our knowledge, the problem described in this paper has not been addressed before. The closest related work is [7] that extracts seven frames from a single motion blurred image. Thus, to obtain a sharp slow-motion video one could apply their method independently to each blurry frame. This approach however faces two issues: first, temporal ordering within each group of seven frames has to be determined, and second, temporal smoothness between intra-/inter-group frames is not guaranteed.

3. Video Slow-Motion and Deblurring

The problem of extracting a sharp video in slow-motion given a blurry video entails two aspects: one is image deblurring and the other is temporal upsampling. We solve the first aspect through a deblurring neural network, which takes as input multiple blurry frames and generates a few sharp frames. We handle the second aspect through an interpolation neural network that generates an intermediate frame between two sharp input frames (for example, produced by the deblurring network). The presence of motion blur in the input frames may seem only a nuisance as it complicates the task of generating a high-quality video in slow-motion. However, motion blur carries information about motion in a video, although in an ambiguous form, and this is quite useful in the extraction of accurate and realistic slow-motion. Therefore, when we use the interpolation network, we feed as input not only two sharp frames, but also the blurry inputs from which the sharp frames are extracted. The fact that the interpolating network still benefits from the blurry frames may suggest that one can solve the overall task with a single feedforward neural network. This solution, however, cannot ensure temporal smoothness. In fact, as the network generates a new output sequence, its first frame may not show a smooth temporal transition from the last frame generated by the previous processing. This is due to the fact that the network has no complete knowledge of the past processing. This problem remains even if we feed multiple blurry images as input to the network, because this input would change from one processing to the next. A possible solution is to use a recurrent neural network, which could store the past in its state. However, the training of recurrent neural networks to generate videos is extremely challenging. Therefore, we propose to approximate the recurrent approach by unfolding and distributing the extraction of the frames over several processing stages. In our architecture intermediate outputs from current and past inputs are combined together to generate the final output (see "frame across inputs" in Fig. 1(b)). This step is fundamental in ensuring the temporal smoothness in our method.

3.1. Image Formation and Notation

We approximate a blurry frame as a discrete averaging process during the exposure time, as already done in [7, 16, 24]. Let τ be the number of discretized sharp frames between two blurry image captures. Let also $\tau - \Delta$ be the number of sharp frames during which the camera aperture is open for one capture, and Δ when the aperture is closed. Then, we denote a sharp frame as S_i^t, where $t = 1, \dots, \tau$ indicates the sharp frames within a capture and i indicates the corresponding captured blurry image. Finally, we can introduce the blurry frame $B_i = \frac{1}{\tau-\Delta} \sum_{t=1}^{\tau-\Delta} S_i^t$. With this notation we call *gap frames* the frames S_i^t with $t = \tau - \Delta + 1, \dots, \tau$. In our method we choose $\tau = 10$ and $\Delta = 1$. Therefore, we have only one gap frame S_i^{10} for each i-th blurry image B_i.

3.2. Problem Statement

Our task is to retrieve the sharp frames S_i^t with $t = 1, \dots, 10$ from the blurry images B_i. In [7] this task was solved by mapping a single blurry image to the corresponding sharp frames. However, this mapping is ambiguous as the temporal ordering is unknown. To address this ambiguity we use multiple blurry images as input. One option is to use two consecutive blurry images B_{1+i}, B_{2+i}. In this case if we choose $\{\hat{S}_{1+i}^t\}_{t=5,\dots,10}, \{\hat{S}_{2+i}^t\}_{t=1,\dots,4}$ as output, then the network must learn to exploit both blurry images B_{1+i}, B_{2+i} and to focus on the temporal transitions between the two inputs. Another option is to use more blurry images, as they provide more information. In fact, we found that using four consecutive blurry images was a good computational/accuracy tradeoff for the current network architectures. Thus, we describe our task as that of extracting frames $\{\hat{S}_{2+i}^t\}_{t=5,\dots,10}, \{\hat{S}_{3+i}^t\}_{t=1,\dots,4}$ from blurry images $B_{1+i}, B_{2+i}, B_{3+i}, B_{4+i}$ and for any i. For simplicity, we use the following more compact notation for the input images $\mathbf{B}_i \doteq (B_{1+i}, B_{2+i}, B_{3+i}, B_{4+i})$.

Figure 2: Ablation study. From left to right: blurry video, cropped region of the blurry video, video results from [7], naïve, constrained, TN and TNTT approaches. **Full version with videos can be found on the project page[1].**

Figure 3: Results on real data with different output frame rates. 5x times results are from (only DeblurNet), 10x are from (DeblurNet + InterpNet), and 20x are from (DeblurNet+InterpNet+InterpNet). **Full version with videos can be found on the project page.**

3.3. Methods

In this section, we discuss several potential formulations and network architectures for our task, and finally introduce the proposed method. In the experiments, we compare these alternative methods via ablation studies. To avoid error accumulation we focus on end-to-end approaches.

Naïve Approach. A straightforward approach is to estimate all the output frames at once. We use the ℓ_1-loss between the predicted outputs and the ground truth. In this approach, we consider training a single network ρ. More precisely, ρ_j, with $j = 1, \ldots, 10$, represents the j-th output of the network. The loss function for a single video is defined as

$$\mathcal{L}_{\text{naïve}} = \sum_i \sum_{j=1}^{10} \left| \check{e}_j^i \right|_1, \tag{1}$$

where i indicates the index of the video frame, and $\check{e}_j^i$ is the error between the prediction and the ground truth

$$\check{e}_j^i = \begin{cases} \rho_j\left(\mathbf{B}_i\right) - S_{2+i}^{j+4}, & j = 1, \ldots, 6 \\ \rho_j\left(\mathbf{B}_i\right) - S_{3+i}^{j-6}, & j = 7, \ldots, 10. \end{cases}$$

By training on this loss function, the model is able to achieve a better performance than applying state-of-the-art deblurring and interpolation sequentially. However, there are two main limitations of this approach. First, the output frame rate cannot be changed after training. Second, there is no guarantee that the output frames are temporally smooth, because $\hat{S}_{3+i}^4$ and $\hat{S}_{3+i}^5$ are estimated independently from the two inputs $\mathbf{B}_i$ and $\mathbf{B}_{i+1}$. In Fig. 2, we show that the naïve approach introduces flickering artifacts.

Constrained Approach. To achieve temporal smoothness, an intuitive idea is to predict overlapping frames from consecutive inputs and constrain them to match. To this aim we train ρ so that it outputs instead 11 frames, with an extra $\hat{S}_{3+i}^5$ from the input $\mathbf{B}_i$. By doing this, $\hat{S}_{3+i}^5$ will be extracted from both the inputs $\mathbf{B}_i$ and $\mathbf{B}_{i+1}$. We then introduce a new loss term that imposes the similarity between $\hat{S}_{3+i}^5 = \hat{e}_{11}^i + S_{3+i}^5$ from $\mathbf{B}_i$ and $\hat{S}_{3+i}^5 = \hat{e}_1^{i+1} + S_{3+i}^5$ from $\mathbf{B}_{i+1}$, *i.e.*,

$$\mathcal{L}_{\text{constrained}} = \sum_i \left| \hat{e}_{11}^i - \hat{e}_1^{i+1} \right|_1 + \sum_{j=1}^{11} \left| \hat{e}_j^i \right|_1 + \left| \hat{e}_j^{i+1} \right|_1 \tag{2}$$

where

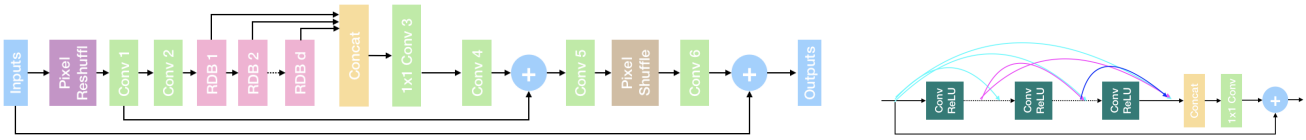
$$\hat{e}_j^i = \begin{cases} \rho_j\left(\mathbf{B}_i\right) - S_{2+i}^{j+4}, & j = 1, \ldots, 6 \\ \rho_j\left(\mathbf{B}_i\right) - S_{3+i}^{j-6}, & j = 7, \ldots, 11. \end{cases}$$

We find experimentally that this approach is not very effective in encouraging temporal smoothness (see Fig. 2). To ensure this smoothness at run time, the network ρ would need to know the future output, but this is not available.

Proposed Approach. In our approach we split the extraction of sharp frames via two neural networks: the DeblurNet, which we denote with ϕ_j, with $j = 1, \ldots, 5$ and outputs five sharp keyframes, and the InterpNet, which we denote with ψ and outputs the frame between two generated keyframes. The key idea is not to extract all output frames simultaneously, but rather to allow some frames to be extracted conditionally on others. This delay allows us to build a smoother transition between frames generated from subsequent inputs even at run time. More precisely, given the input $\mathbf{B}_i$, DeblurNet outputs five keyframes $\hat{S}_{2+i}^5$, $\hat{S}_{2+i}^7$, $\hat{S}_{2+i}^9$, $\hat{S}_{3+i}^1$ and $\hat{S}_{3+i}^3$ (*i.e.*, the odd-indexed sharp frames). Afterwards, InterpNet extracts frames $\hat{S}_{2+i}^6$, $\hat{S}_{2+i}^8$, $\hat{S}_{2+i}^{10}$, and $\hat{S}_{3+i}^2$ (*i.e.*, the even-indexed sharp frames) conditioned on the outputs of DeblurNet. We then define the loss function for a single video as in the naïve approach as

$$\mathcal{L}_{\text{proposed}} = \sum_i \sum_{j=1}^{10} \left| e_j^i \right|_1, \tag{3}$$

[1] https://github.com/MeiguangJin/slow-motion



(a) (b)

Figure 4: (a) Architecture of DeblurNet and (b) residual dense block.

Figure 5: Gap frame interpolation comparison.

where the errors are defined as

$$\begin{cases} e^i_{2k-1} = \phi_k(\mathbf{B}_i) - S^{2k+3}_{2+i}, & k = 1,2,3 \\ e^i_{2k-1} = \phi_k(\mathbf{B}_i) - S^{2k-7}_{3+i}, & k = 4,5 \\ e^i_{2k} = \psi(B_{2+i}, \phi_k(\mathbf{B}_i), \phi_{k+1}(\mathbf{B}_i), B_{3+i}) - S^{2k+4}_{2+i}, & k = 1,2,3 \\ e^i_{2k} = \psi(B_{2+i}, \phi_k(\mathbf{B}_i), \phi_{k+1}(\mathbf{B}_i), B_{3+i}) - S^{2k-6}_{3+i}, & k = 4 \\ e^i_{2k} = \psi\left(B_{2+i}, \phi_k(\mathbf{B}_i), \phi_1(\mathbf{B}_{i+1}), B_{3+i}\right) - S^{2k-6}_{3+i}, & k = 5. \end{cases}$$

DeblurNet takes as input the blurry frames $\mathbf{B}_i$ as before. However, now also InterpNet takes as input blurry frames. More precisely, it uses the blurry frames B_{2+i} and B_{3+i}, which directly relate to all the outputs. Moreover, the last error e^i_{10} is the term that encourages temporal smoothness during training. The overall model is shown in Fig. 1.

The proposed training improves significantly over the naïve and constrained approaches, as shown quantitatively in Tables 1 and 2 and qualitatively in Fig. 2. To show that training with the error e^i_{10} is necessary to avoid flickering artifacts, we distinguish two versions of our method. We call Two-Network with Temporal-Transition (**TNTT**) the case where e^i_{10} is used in the loss function and simply Two-Network (**TN**) the case without e^i_{10}. The proposed architecture allows also to increase the frame rate at run-time by applying InterpNet between pairs of adjacent frames of the previous output sequence. This allows to double the frame rate by increasing the number of computations. Because frames at lower frame rates are available sooner than those at higher frame rates, this arrangement appeals to systems that render videos with flexible frame rates. We demonstrate our approach with a $5\times$ frame rate increase (only DeblurNet), a $10\times$ frame rate increase (DeblurNet + InterpNet), and a $20\times$ frame rate increase (DeblurNet + InterpNet + InterpNet) in Fig. 3.

3.4. Model Architecture and Loss Functions

Both DeblurNet and InterpNet are feedforward convolutional neural networks, and we adopt an architecture similar to that used in the recent super-resolution work [29]. The architecture of DeblurNet (see Fig. 4a) employs several residual dense blocks (RDB) [29] (see Fig. 4b), which exploit full hierarchical features from all the convolutional layers. To handle interpolation and deblurring tasks with large motion, the network requires a wide receptive field. Towards this purpose, we first use a *pixel reshuffle* layer that rearranges tensor elements between the spatial and channel coordinates, as was done in [7], and later on its inverse layer, *pixel shuffle*, also called *sub-pixel convolution* in [23]. InterpNet shares the same structure as DeblurNet and only differs in the number of RDBs and the number of outputs. DeblurNet includes 20 RDBs and predicts five frames, while InterpNet includes 10 RDBs and estimates one frame. Except for the first convolutional layer using 5×5 kernels and the convolutional layer after the concat layer using 1×1 kernels, all other convolutional layers use 3×3 kernels in our networks. In both networks, the number of feature channels are 128, the *growth rate* for the RDBs is set to 48 and 5 convolution layers are used in each RDB.

3.5. Training Data

We find empirically that training using the data generated by averaging generalizes well to real data. Thus, we collect a new dataset with 40 high-quality videos at 250 FPS from a Sony RX V camera, and each video contains 1000 frames at 1080P. To the best of our knowledge, this dataset is the largest high-quality, high-resolution and high-frame-rate video dataset currently available. To avoid domain bias towards the capturing device during training, we include 20 GoPro 720P videos at 240 FPS from the work [16], where each video contains 900 frames. During the training, we synthesize the blurry inputs on the fly such that all frames can be used for training. For a qualitative comparison, we use a separate test set including 5 GoPro 240 FPS videos

Figure 6: Comparisons on real data: Blurry videos are captured from our sony camera (Full HD video). **Full version with videos can be found on the project page.**

from [16], 4 Sony RX V 250 FPS videos that we captured and real blurry videos captured with a Sony camera under 25 FPS in Full HD mode.

3.6. Implementation Details

The proposed method is implemented with PyTorch. We use a batch-size of 4 and a patch-size of 224×224 pixels for training. Samples are augmented by random rotation and flipping, as well as adding 1% Gaussian noise. We use the Adam optimizer [12] and start with a learning rate of $5 \cdot 10^{-5}$. The learning rate is divided by 2 after every 20K iterations and we run 100K iterations in total. The whole training takes about 2 days with two TITAN X GPUs.

4. Experiments

In this section, we carry out a number of experiments to evaluate both quantitatively and qualitatively the deblurring/interpolation performance of our network. For the quantitative comparison, we select 9 videos, of which five are GoPro videos at 240FPS from [16], and four are from our own Sony RX V camera captured at 250FPS. #1 to #5 in Tables 1 and 2 denote the GoPro videos, #6 to #9 are the Sony videos. Each video contains 400 sharp frames, and we average 9 frames to synthesize a blurry input. Between two blurry frames we discard one sharp frame. Hence, all together we generate 40 blurry frames from each video.

Deblurring. Because our network is able to both deblur the input blurry frames and interpolate between them, we evaluate the network deblurring performance separately from its deblurring+interpolation performance. For deblurring, we compare with the state of the art single image deblurring methods [13, 16] and a video deblurring approach [24], which uses five blurry inputs. Since we use four input frames, while [13, 16] use a single input frame, we take [13, 16] as baseline methods to improve upon. PSNR evaluation is shown for each video in Table 1. It can be seen that our network performs much better than the other three approaches on both datasets. Two synthetic and real comparisons are shown in Fig. 8 for a qualitative evaluation.

Slow Motion and Deblurring. Our network is able to simultaneously deblur and interpolate nine frames given four blurry inputs. We quantitatively evaluate the accuracy of these nine reconstructed frames. Because this is a novel problem, there are no existing algorithms that we can compare to. To better emphasize our contributions we use as alternative approaches the combinations of existing de-

Table 1: Deblurring performance on the middle sharp frames in terms of PSNR .

Method\Video	#1	#2	#3	#4	#5	#6	#7	#8	#9
Blurry	31.8	31.9	26.8	25.5	22.9	36.5	27.1	32.8	**39.2**
Nah [16]	33.0	32.4	27.7	26.2	26.0	35.6	28.0	29.9	35.6
Su [24]	33.2	32.5	28.0	27.0	26.0	35.9	29.4	32.1	34.4
Kupyn [13]	31.6	31.4	26.5	25.3	23.4	33.5	23.8	32.2	34.8
Naïve	**33.9**	34.5	29.3	27.9	**26.3**	**38.3**	31.3	35.4	38.6
TN	**33.9**	34.5	**29.4**	27.9	**26.3**	**38.3**	31.3	**35.5**	38.7
TNTT	**33.9**	**34.6**	29.3	**28.0**	**26.3**	**38.3**	**31.4**	**35.5**	38.7

blurring and interpolation methods. We evaluate the performance of two schemes, interpolation after deblurring and deblurring after interpolation. We consider the high-performance methods [13], [16] and [24] for deblurring. For interpolation, we consider two approaches: One is to apply a `flow`-based approach from [24], and the other is the state of the art video interpolation method [6]. Essentially, both interpolation approaches involve optical flow estimates between two frames, thus generating any possible slow motion video. This enables a frame to frame reconstruction comparison with TNTT. By combining the three deblurring approaches [13], [16] and [24] with the two interpolation techniques under the two schemes (pre/post deblurring), we evaluate 12 different video reconstruction results. All results are shown in Table 2. We observe that TNTT can perform better than other alternative approaches. Notice that in many cases, a two-step approach tends to accumulate artifacts in the first step and degrade the video quality. This might yield a performance that is worse than simply interpolating the input blurry frames. To ignore the artifacts from the deblurring step, we also show the interpolation performance of applying the flow approach to the ground truth sharp frames. Results are shown in the last second block of Table 2. We can see that interpolation with ground truth frames outperforms all the other alternative approaches. However, TNTT is still better. This is due to the fact that our InterpNet can make use of motion information in the blurry inputs. This also shows empirically that solving the problem in two separate steps (deblurring and then interpolation) is sub optimal, even when the ground truth (for deblurring) is given. For a qualitative evaluation, we show the interpolated results of the gap frame in Fig. 5.

Ablation Study. To see the effectiveness of our design choices, we evaluate our network and other design choices both quantitatively and qualitatively. In Table 1, we see that both naïve and TN methods achieve a very close performance to TNTT in the deblurring of the center frame. However, TNTT outperforms the other two approaches consistently in the interpolation evaluation in Table 2. Since optical flow is an indicator of motion, we also measure the temporal smoothness of our output videos with optical flow. We apply a flow estimation algorithm to three consecutive

Table 2: Interpolation performance on 9 interpolated frames in terms of PSNR .

Method\Video	#1	#2	#3	#4	#5	#6	#7	#8	#9
Blurry + Jiang [6]	29.7	29.9	25.1	24.0	22.1	35.2	26.7	32.3	36.3
Blurry + flow	29.5	29.3	24.9	23.8	22.0	34.6	26.6	32.1	35.4
Nah [16]+ Jiang [6]	29.9	30.1	25.1	23.9	24.2	34.6	27.9	30.0	34.4
Su [24]+ Jiang [6]	30.7	30.5	26.0	25.1	24.0	35.0	29.2	32.0	33.4
Kupyn [13]+ Jiang [6]	30.1	30.1	25.3	24.2	22.8	33.0	24.0	32.1	34.0
Jiang [6]+ Nah [16]	30.6	30.6	25.8	24.5	24.1	34.5	27.7	29.8	34.3
Jiang [6]+ Su [24]	30.2	30.1	25.6	24.4	23.4	34.8	28.4	32.4	33.7
Jiang [6]+ Kupyn [13]	29.7	29.8	25.2	24.1	22.5	32.6	23.8	31.7	33.5
Nah [16]+ flow	30.6	30.3	25.7	24.5	24.1	34.4	27.9	30.1	34.2
Su [24]+ flow	30.6	30.2	25.9	24.9	24.0	34.6	29.1	32.0	33.2
Kupyn [13]+ flow	29.1	28.1	24.8	23.8	22.4	31.8	23.9	31.4	31.7
flow + Nah [16]	29.9	29.9	25.2	24.0	23.5	34.0	27.6	29.7	33.6
flow + Su [24]	29.6	29.4	25.1	24.2	23.1	34.2	28.2	32.2	33.3
flow + Kupyn [13]	29.3	29.1	24.7	23.7	22.4	32.4	23.7	31.1	33.0
GT + flow	31.1	31.3	26.9	26.0	25.9	36.1	31.7	34.9	35.8
Naïve	32.3	32.9	28.5	27.0	26.2	37.4	31.2	35.3	36.4
TN	32.2	33.1	28.4	27.1	25.8	37.4	31.0	35.3	36.3
TNTT	**32.4**	**33.2**	**28.8**	**27.5**	**26.9**	**37.5**	**31.8**	**35.5**	**36.5**

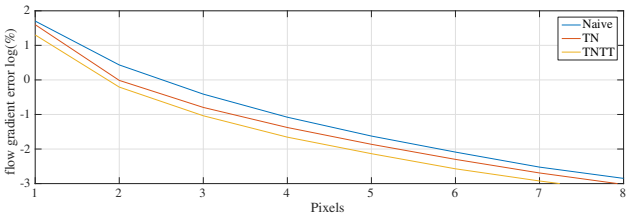


Figure 7: Temporal smoothness comparison. We compute the optical flow between frames predicted from the TNTT, naïve, TN methods, and the ground truth. On the ordinate axis we plot the number of optical flow gradient matching errors larger than a threshold in logarithmic scale.

ground truth frames to get the flow gradient and use it as a reference. Similarly we do the same calculation to frames predicted from the TNTT, naïve and TN methods. Then, we match the flow gradient to the reference flow gradient. We count the number of flow gradient matching errors larger than a threshold (in the range $1, \ldots, 8$ pixels) and plot the percentage in logarithmic scale. Results are shown in Fig. 7. We observe that the TN method preserves smoothness better than the naïve approach and that the TNTT method achieves the best performance. We also show a qualitative comparison of these three approaches in Fig. 2.

Real Comparison. To see the generalization capability of our network, we evaluate our approach on real blurry videos captured with our Sony camera. Notice that real blurry videos are Full HD low frame rate videos and they are coded differently from our training dataset. We combine the three deblurring algorithms [13], [16] and [24] with two interpolation algorithms and show all these results in Fig. 6. One can observe that our model generates a more realistic video than other methods especially around the car wheels. More comparisons can be found in the supplementary material.

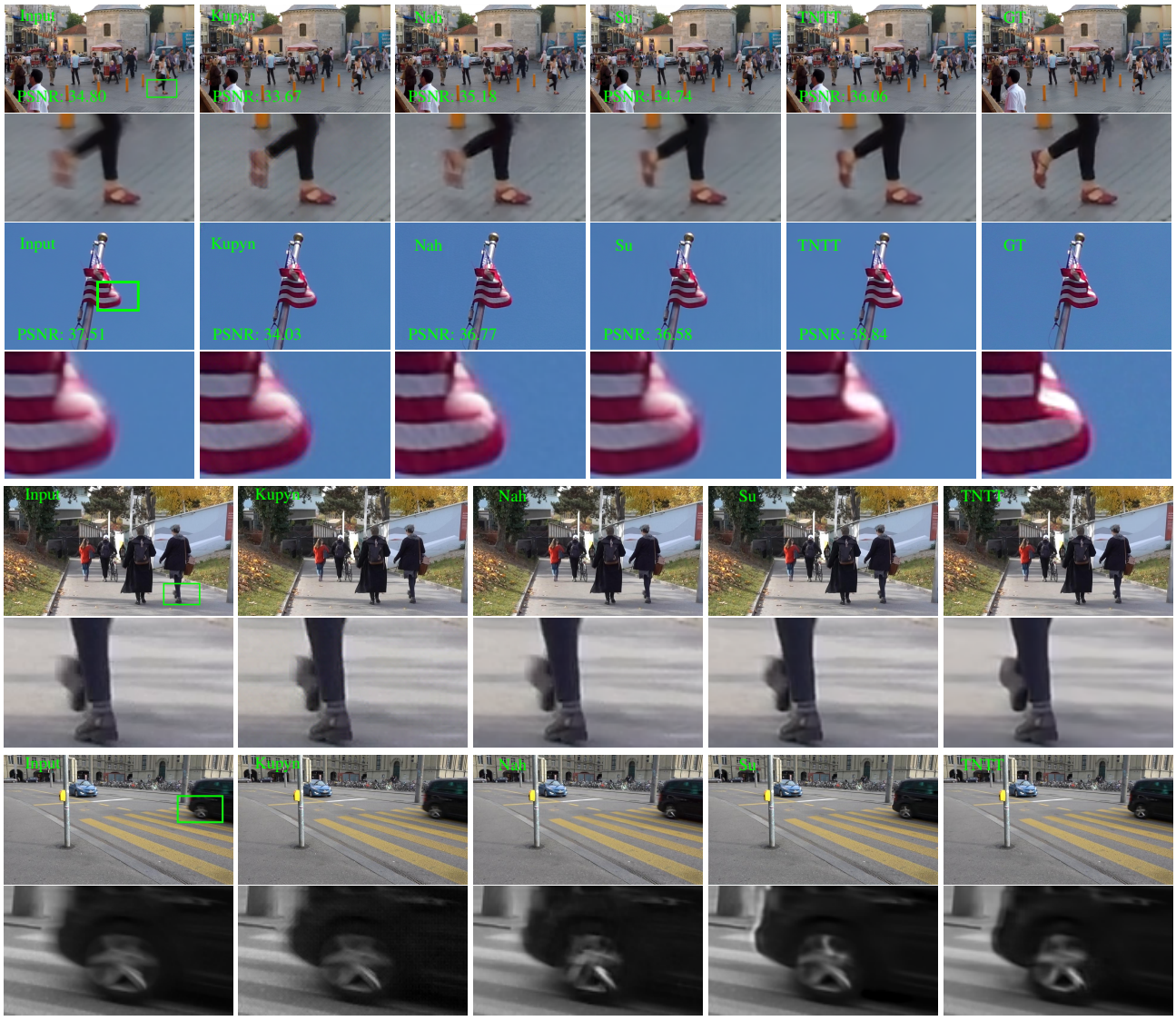


Figure 8: Qualitative comparison: From the left column to the right column: the blurry input, deblurring results of [13], [16], [24], TNTT and ground truth. The first and second rows are from the GoPro test set [16] and the third and fourth rows are from our Sony test set. The fifth to eighth rows show two real comparisons captured with our Sony camera (Full HD video).

Slow Motion at Different Frame Rates. As mentioned in the previous section, DeblurNet extracts 5-fold slow motion videos. To demonstrate the effectiveness of our InterpNet, we recursively apply it and generate 10-fold and 20-fold slow motion videos. We test on a real blurry video and results are shown in Fig. 3. One can observe that our InterpNet can generate a realistic 20-fold slow motion video.

Limitations. The main limitation of our approach is that our model is not robust to very large blurs. This is a common challenge for deblurring networks [7]. In this case, predicted videos will show flickering artifacts. However, in this case our model is able to obtain a better deblurring accuracy than other state of the art deblurring methods.

5. Conclusions

In this paper, we have presented the first method to generate a sharp slow-motion video from a low frame rate blurry video. We have shown that the main challenge of this task is to preserve the temporal smoothness. We have presented an approach based on two networks, which can not only address the temporal smoothness issue, but also increase the frame rate indefinitely. We have demonstrated that our model can successfully extract slow motion videos on both synthetic and real blurry videos.

Acknowledgements. MJ and PF acknowledge support from the Swiss National Science Foundation on project 200021_153324.

References

[1] Sunghyun Cho and Seungyong Lee. Fast motion deblurring. *ACM Transactions on Graphics*, 2009. 2

[2] Sunghyun Cho, Jue Wang, and Seungyong Lee. Video deblurring for hand-held cameras using patch-based synthesis. *ACM Transactions on Graphics*, 2012. 2

[3] Robert Fergus, Barun Singh, Aaron Hertzmann, Sam T. Roweis, and William T. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics*, 2006. 2

[4] Michal Hradiš, Jan Kotera, Pavel Zemcík, and Filip Šroubek. Convolutional neural networks for direct text deblurring. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2015. 2

[5] Zhe Hu, Li Xu, and Ming-Hsuan Yang. Joint depth estimation and camera shake removal from single blurry image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[6] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 3, 5, 6, 7

[7] Meiguang Jin, Givi Meishvili, and Paolo Favaro. Learning to extract a video sequence from a single motion-blurred image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 3, 4, 5, 8

[8] Meiguang Jin, Stefan Roth, and Paolo Favaro. Normalized blind deconvolution. In *The European Conference on Computer Vision (ECCV)*, 2018. 2

[9] Tae Hyun Kim, Byeongjoo Ahn, and Kyoung Mu Lee. Dynamic scene deblurring. In *The IEEE International Conference on Computer Vision (ICCV)*, 2013. 2

[10] Tae Hyun Kim, Kyoung Mu Lee, Bernhard Schölkopf, and Michael Hirsch. Online video deblurring via dynamic temporal blending network. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[11] Tae Hyun Kim, Seungjun Nah, and Kyoung Mu Lee. Dynamic scene deblurring using a locally adaptive linear blur model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. 2

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

[13] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 5, 6, 7, 8

[14] Ziwei Liu, Raymond Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 3

[15] Dhruv Mahajan, Fu-Chung Huang, Wojciech Matusik, Ravi Ramamoorthi, and Peter Belhumeur. Moving gradients: a path-based method for plausible image interpolation. In *ACM Transactions on Graphics*, 2009. 3

[16] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 3, 5, 6, 7, 8

[17] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 3

[18] T M Nimisha, Akash K Singh, and Ambasamudram N Rajagopalan. Blur-invariant deep learning for blind-deblurring. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[19] Mehdi Noroozi, Paramanand Chandramouli, and Paolo Favaro. Motion deblurring in the wild. In *Proceedings of the German Conference on Pattern Recognition (GCPR)*, 2017. 2

[20] Jinshan Pan, Deqing Sun, Hanspeter Pfister, and Ming-Hsuan Yang. Blind image deblurring using dark channel prior. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[21] Chandramouli Paramanand and Ambasamudram N Rajagopalan. Non-uniform motion deblurring for bilayer scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2

[22] Qi Shan, Jiaya Jia, and Aseem Agarwala. High-quality motion deblurring from a single image. *ACM Transactions on Graphics*, 2008. 2

[23] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[24] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 3, 5, 6, 7, 8

[25] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[26] Oliver Whyte, Josef Sivic, Andrew Zisserman, and Jean Ponce. Non-uniform deblurring for shaken images. *International Journal of Computer Vision (IJCV)*, 2012. 2

[27] Li Xu and Jiaya Jia. Two-phase kernel estimation for robust motion deblurring. In *The European Conference on Computer Vision (ECCV)*, 2010. 2

[28] Li Xu, Shicheng Zheng, and Jiaya Jia. Unnatural L_0 sparse representation for natural image deblurring. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2

[29] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[30] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow.

In *The European Conference on Computer Vision (ECCV)*, 2016. 3

[31] C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics*, 2004. 3