Detail-revealing Deep Video Super-resolution*

Xin Tao[1] Hongyun Gao[1] Renjie Liao[2,3] Jue Wang[4] Jiaya Jia[1,5]
[1]The Chinese University of Hong Kong [2]University of Toronto
[3]Uber Advanced Technologies Group [4]Megvii Inc. [5]Youtu Lab, Tencent
{xtao,hygao}@cse.cuhk.edu.hk rjliao@cs.toronto.edu
arphid@gmail.com leojia9@gmail.com

Abstract

Previous CNN-based video super-resolution approaches need to align multiple frames to the reference. In this paper, we show that proper frame alignment and motion compensation is crucial for achieving high quality results. We accordingly propose a "sub-pixel motion compensation" (SPMC) layer in a CNN framework. Analysis and experiments show the suitability of this layer in video SR. The final end-to-end, scalable CNN framework effectively incorporates the SPMC layer and fuses multiple frames to reveal image details. Our implementation can generate visually and quantitatively high-quality results, superior to current state-of-the-arts, without the need of parameter tuning.

1. Introduction

As one of the fundamental problems in image processing and computer vision, video or multi-frame super-resolution (SR) aims at recovering high-resolution (HR) images from a sequence of low-resolution (LR) ones. In contrast to single-image SR where details have to be generated based on only external examples, an ideal video SR system should be able to correctly extract and fuse image details in multiple frames. To achieve this goal, two important sub-problems are to be answered: (1) how to align multiple frames to construct accurate correspondence; and (2) how to effectively fuse image details for high-quality outputs.

Motion Compensation While large motion between consecutive frames increases the difficulty to locate corresponding image regions, subtle sub-pixel motion contrarily benefits restoration of details. Most previous methods compensate inter-frame motion by estimating optical flow [2, 7, 19, 20, 23] or applying block-matching [28]. After motion is estimated, traditional methods [7, 20, 23] reconstruct the HR output based on various imaging models and image priors, typically under an iterative estimation framework. Most of these methods involve rather intensive case-by-case parameter-tuning and costly computation.

Recent deep-learning-based video SR methods [2, 14] compensate inter-frame motion by aligning all other frames to the reference one, using backward warping. We show that such a seemingly reasonable technical choice is actually not optimal for video SR, and improving motion compensation can directly lead to higher quality SR results. In this paper, we achieve this by proposing a sub-pixel motion compensation (SPMC) strategy, which is validated by both theoretical analysis and extensive experiments.

Detail Fusion Besides motion compensation, proper image detail fusion from multiple frames is the key to the success of video SR. We propose a new CNN framework that incorporates the SPMC layer, and effectively fuses image information from aligned frames. Although previous CNN-based video SR systems can produce sharp-edge images, it is not entirely clear whether the image details are those inherent in input frames, or learned from external data. In many practical applications such as face or text recognition, only true HR details are useful. In this paper we provide insightful ablation study to verify this point.

Scalability A traditionally-overlooked but practically-meaningful property of SR systems is the scalability. In many previous learning-based SR systems, the network structure is closely coupled with SR parameters, making them less flexible when new SR parameters need to be applied. For example, ESPCN [26] output channel number is determined by the scale factor. VSRnet [14] and VESPCN [2] can only take a fixed number of temporal frames as input, once trained.

In contrast, our system is fully scalable. First, it can take arbitrary-size input images. Second, the new SPMC layer does not contain trainable parameters and can be applied for arbitrary scaling factors during testing. Finally, the ConvLSTM-based [29] network structure makes it possible to accept an arbitrary number of frames for SR in testing

*This work is in part supported by a grant from the Research Grants Council of the Hong Kong SAR (project No. 413113).

phase.

1.1. Related Work

Deep Super-resolution With the seminal work of SRCNN [3], a majority of recent SR methods employ deep neural networks [4, 11, 15, 16, 18, 26]. Most of them resize input frames before sending them to the network [4, 11, 15, 16], and use very deep [15], recursive [16] or other networks to predict HR results. Shi *et al*. [26] proposed a subpixel network, which directly takes low-resolution images as input, and produces a high-res one with subpixel location. Ledig *et al*. [18] used a trainable deconvolution layer instead.

For deep video SR, Liao *et al*. [19] adopted a separate step to construct high-resolution SR-drafts, which are obtained under different flow parameters. Kappeler *et al*. [14] estimated optical flow and selected corresponding patches across frames to train a CNN. In both methods, motion estimation is separated from training. Recently, Caballero *et al*. [2] proposed the first end-to-end video SR framework, which incorporates motion compensation as a submodule.

Motion Estimation Deep neural networks were also used to solve motion estimation problems. Zbontar and LeCun [31] and Luo *et al*. [22] used CNNs to learn a patch distance measure for stereo matching. Fischer *et al*. [8] and Mayer *et al*. [25] proposed end-to-end networks to predict optical flow and stereo disparity.

Progress was made in spatial transformer networks [10] where a differentiable layer warps images according to predicted affine transformation parameters. Based on it, WarpNet [13] used a similar scheme to extract sparse correspondence. Yu *et al*. [30] warped output based on predicted optical flow as a photometric loss for unsupervised optical flow learning. Different from these strategies, we introduce a *Sub-pixel Motion Compensation* (SPMC) layer, which is suitable for the video SR task.

2. Sub-pixel Motion Compensation (SPMC)

We first introduce our notations for video SR. It takes a sequence of $N_F = (2T+1)$ LR images as input (T is the size of temporal span in terms of number of frames), where $\Omega_L = \{I^L_{-T}, \cdots, I^L_0, \cdots, I^L_T\}$. The output HR image I^H_0 corresponds to center reference frame I^L_0.

LR Imaging Model The classical imaging model for LR images [7, 19, 20, 23] is expressed as

$$I^L_i = SKW_{0\to i}I^H_0 + n_i, \tag{1}$$

where $W_{0\to i}$ is the warping operator to warp from the 0th to ith frame. K and S are downsampling blur and decimation operators, respectively. n_i is the additive noise to frame i. For simplicity's sake, we neglect operator K in the following analysis, since it can be absorbed by S.

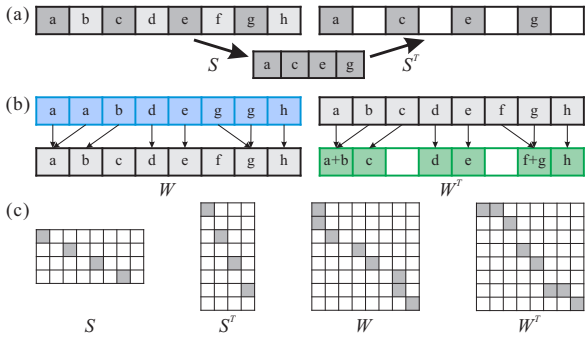


Figure 1. **Visualization of operators in image formation**. (a) Decimation operator S ($2\times$) reduces the input 1D signal to its half-size. The transpose S^T corresponds to zero-upsampling. (b) With arrows indicating motion, warping operator W produces the **blue** signal from the **gray** one through backward warping. W^T produces the **green** signal through forward warping. (c) Illustration of matrices S, S^T, W and W^T. Grayed and white blocks indicate values 1 and 0 respectively.

Flow Direction and Transposed Operators Operator $W_{0\to i}$ indicates the warping process. To compute it, one needs to first calculate the motion field $F_{i\to 0}$ (from the ith to 0th frame), and then perform backward warping to produce the warped image. However, current deep video SR methods usually align other frames back to I^L_0, which actually makes use of flow $F_{0\to i}$.

More specifically, directly minimizing the L_2-norm reconstruction error $\sum_i \|SW_{0\to i}I^H_0 - I^L_i\|^2$ results in

$$I^H_0 = (\sum_i W^T_{0\to i}S^T S W_{0\to i})^{-1}(\sum_i W^T_{0\to i}S^T I^L_i). \tag{2}$$

With certain assumptions [5, 7], $W^T_{0\to i}S^T S W_{0\to i}$ becomes a diagonal matrix. The solution to Eq. (2) reduces to a feed-forward generation process of

$$I^H_0 = \frac{\sum_i W^T_{0\to i}S^T I^L_i}{\sum_i W^T_{0\to i}S^T \mathbf{1}}, \tag{3}$$

where $\mathbf{1}$ is an all-one vector with the same size as I^L_i. The operators that are actually applied to I^L_i are S^T and $W^T_{0\to i}$. S^T is the transposed decimation corresponding to zero-upsampling. $W^T_{0\to i}$ is the transposed forward warping using flow $F_{i\to 0}$. A 1D signal example for these operators is shown in Fig. 1. We will further analyze the difference of forward and backward warping after explaining our system.

3. Our Method

Our method takes a sequence of N_F LR images as input and produces one HR image I^H_0. It is an end-to-end fully trainable framework that comprises of three modules: motion estimation, motion compensation and detail fusion.

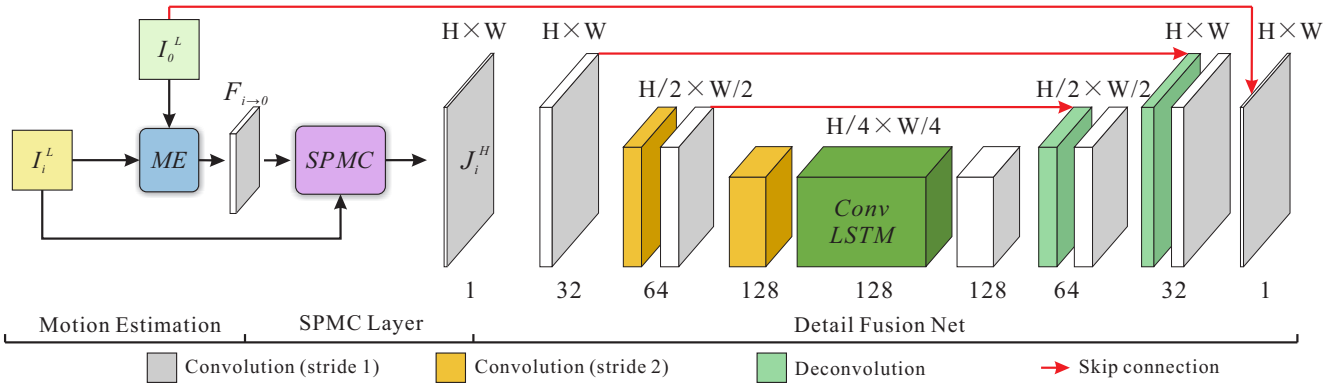


Figure 2. **Our framework.** Network configuration for the ith time step.

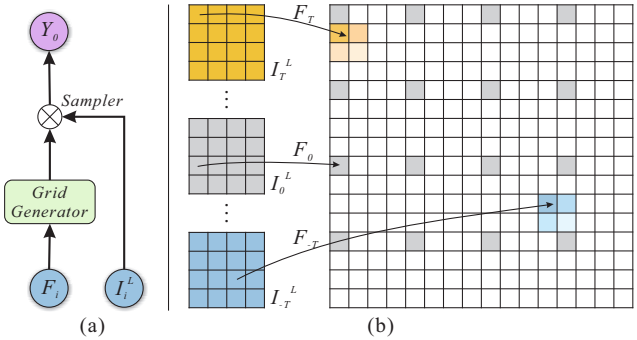


Figure 3. **Subpixel Motion Compensation layer** ($\times 4$). (a) Layer diagram. (b) Illustration of the SPMC layer ($\times 4$).

They are respectively responsible for motion field estimation between frames; aligning frames by compensating motion; and finally increasing image scale and adding image details. We elaborate on each module in the following.

3.1. Motion Estimation

The motion estimation module takes two LR frames as input and produces a LR motion field as

$$F_{i\to j} = \mathbf{Net}_{ME}(I_i^L, I_j^L; \theta_{ME}), \tag{4}$$

where $F_{i\to j} = (u_{i\to j}, v_{i\to j})$ is the motion field from frame I_i^L to I_j^L. θ_{ME} is the set of module parameters.

Using neural networks for motion estimation is not a new idea, and existing work [2, 8, 25, 30] already achieves good results. We have tested FlowNet-S [8] and the motion compensation transformer (MCT) module from VESPCN [2] for our task. We choose MCT because it has less parameters and accordingly less computation cost. It can process 500+ single-channel image pairs (100×100 in pixels) per second. The result quality is also acceptable in our system.

3.2. SPMC Layer

According to the analysis in Sec. 2, we propose a novel layer to utilize sub-pixel information from motion and simultaneously achieve sub-pixel motion compensation (SPMC) and resolution enhancement. It is defined as

$$J^H = \mathbf{Layer}_{SPMC}(J^L, F; \alpha), \tag{5}$$

where J^L and J^H are input LR and output HR images, F is optical flow used for transposed warping and α is the scaling factor. The layer contains two submodules.

Sampling Grid Generator In this step, transformed coordinates are first calculated according to estimated flow $F = (u, v)$ as

$$\begin{pmatrix} x_p^s \\ y_p^s \end{pmatrix} = W_{F;\alpha} \begin{pmatrix} x_p \\ y_p \end{pmatrix} = \alpha \begin{pmatrix} x_p + u_p \\ y_p + v_p \end{pmatrix}, \tag{6}$$

where p indexes pixels in LR image space. x_p and y_p are the two coordinates of p. u_p and v_p are the flow vectors estimated from previous stage. We denote transform of coordinates as operator $W_{F;\alpha}$, which depends on flow field F and scale factor α. x_p^s and y_p^s are the transformed coordinates in an enlarged image space, as shown in Fig. 3.

Differentiable Image Sampler Output image is constructed in the enlarged image space according to x_p^s and y_p^s. The resulting image J_q^H is

$$J_q^H = \sum_{p=1} J_p^L M(x_p^s - x_q) M(y_p^s - y_q), \tag{7}$$

where q indexes HR image pixels. x_q and y_q are the two coordinates for pixel q in the HR grid. $M(\cdot)$ is the sampling kernel, which defines the image interpolation methods (*e.g.* bicubic, bilinear, and nearest-neighbor).

We further investigate differentiability of this layer. As indicated in Eq. (5), the SPMC layer takes one LR image J^L and one flow field $F = (u, v)$ as input, without other trainable parameters. For each output pixel, partial deriva-

tive with respect to each input pixel is

$$\frac{\partial J_q^H}{\partial J_p^L} = \sum_{p=1} M(x_p^s - x_q)M(y_p^s - y_q). \tag{8}$$

It is similar to calculating partial derivatives with respect to flow field (u_p, v_p) using the chain rule as

$$\frac{\partial J_q^H}{\partial u_p} = \frac{\partial J_q^H}{\partial x_p^s} \cdot \frac{\partial x_p^s}{\partial u_p} = \alpha \sum_{p=1} J_p^L M'(x_p^s - x_q)M(y_p^s - y_q), \tag{9}$$

where $M'(\cdot)$ is the gradient of sampling kernel $M(\cdot)$. Similar derivatives can be derived for $\frac{\partial J_q}{\partial v_p}$. We choose $M(x) = \max(0, 1 - |x|)$, which corresponds to the bilinear interpolation kernel, because of its simplicity and convenience to calculate gradients. Our final layer is fully differentiable, allowing back-propagating loss to flow fields smoothly. The advantages of having this type of layers is threefold.

- This layer can simultaneously achieve motion compensation and resolution enhancement. Note in most previous work, they are separate steps (*e.g.* backward warping + bicubic interpolation).
- This layer is parameter free and fully differentiable, which can be effectively incorporated into neural networks with almost no additional cost.
- The rationale behind this layer roots from accurate LR imaging model, which ensures good performance in theory. It also demonstrates good results in practice, as we will present later.

3.3. Detail Fusion Net

The SPMC layer produces a series of motion compensated frames $\{J_i^H\}$ expressed as

$$J_i^H = \mathbf{Layer}_{SPMC}(I_i^L, F_{i\to 0}; \alpha). \tag{10}$$

Design of the following network is non-trivial due to the following considerations. First, $\{J_i^H\}$ are already HR-size images that produce large feature maps, thus computational cost becomes an important factor.

Second, due to the property of forward warping and zero-upsampling, $\{J_i^H\}$ is sparse and majority of the pixels are zero-valued (*e.g.* about 15/16 are zeros for scale factor $4\times$). This requires the network to have large receptive fields to capture image patterns in J_i^H. Using simple interpolation to fill these holes is not a good solution because interpolated values would dominate during training.

Finally, special attention needs to be paid to the use of the reference frame. On the one hand, we rely on the reference frame as the guidance for SR so that the output HR image is consistent with the reference frame in terms of image structures. On the other hand, over-emphasizing the reference frame could impose an adverse effect of neglecting information in other frames. The extreme case is that the system behaves like a single-image SR one.

Network Architecture We design an encoder-decoder [24] style structure with skip-connections (see Fig. 2) to tackle above issues. This type of structure has been proven to be effective in many image regression tasks [21, 24, 27]. The encoder sub-network reduces the size of input HR image to $1/4$ of it in our case, leading to reduced computation cost. It also makes the feature maps less sparse so that information can be effectively aggregated without the need of employing very deep networks. Skip-connections are used for all stages to accelerate training.

A ConvLSTM module [29] is inserted in the middle stage as a natural choice for sequential input. The network structure includes

$$\begin{aligned} \mathbf{f}_i &= \mathbf{Net}_E(J_i^H; \theta_E) \\ \mathbf{g}_i, \mathbf{s}_i &= \mathbf{ConvLSTM}(\mathbf{f}_i, \mathbf{s}_{i-1}; \theta_{LSTM}) \\ I_0^{(i)} &= \mathbf{Net}_D(\mathbf{g}_i, \mathbf{S}_i^E; \theta_D) + I_0^{L\uparrow} \end{aligned} \tag{11}$$

where $\mathbf{Net}_E$ and $\mathbf{Net}_D$ are encoder and decoder CNNs with parameters θ_E and θ_D. $\mathbf{f}_i$ is the output of encoder net. $\mathbf{g}_i$ is the input of decoder net. $\mathbf{s}_i$ is the hidden state for LSTM at the ith step. $\mathbf{S}_i^E$ for all i are intermediate feature maps of $\mathbf{Net}_E$, used for skip-connection. $I_0^{L\uparrow}$ is the bicubic upsampled I_0^L. $I_0^{(i)}$ is the ith time step output.

The first layer of $\mathbf{Net}_E$ and the last layer of $\mathbf{Net}_D$ have kernel size 5×5. All other convolution layers use kernel size 3×3, including those inside ConvLSTM. Deconvolution layers are with kernel size 4×4 and stride 2. Rectified Linear Units (ReLU) are used for every conv/deconv layer as the activation function. For skip-connection, we use SUM operator between connected layers. Other parameters are labeled in Fig. 2.

3.4. Training Strategy

Our framework consists of three major components, each has a unique functionality. Training the whole system in an end-to-end fashion with random initialization would result in zero flow in motion estimation, making the final results similar to those of single-image SR. We therefore separate training into three phases.

Phase 1 We only consider $\mathbf{Net}_{ME}$ in the beginning of training. Since we do not have ground truth flow, unsupervised warping loss is used as [21, 30]

$$\mathcal{L}_{ME} = \sum_{i=-T}^{T} \|I_i^L - \tilde{I}_{0\to i}^L\|_1 + \lambda_1 \|\nabla F_{i\to 0}\|_1, \tag{12}$$

where $\tilde{I}_{0\to i}^L$ is the backward warped I_0^L according to estimated flow $F_{i\to 0}$, using a differentiable layer similar to spatial transformer [10]. Note that this image is in low resolution, aligned with I_i^L. $\|\nabla F_{i\to 0}\|_1$ is the total variation term on each (u, v)-component of flow $F_{i\to 0}$. λ_1 is the regularization weight. We set $\lambda_1 = 0.01$ in all experiments.

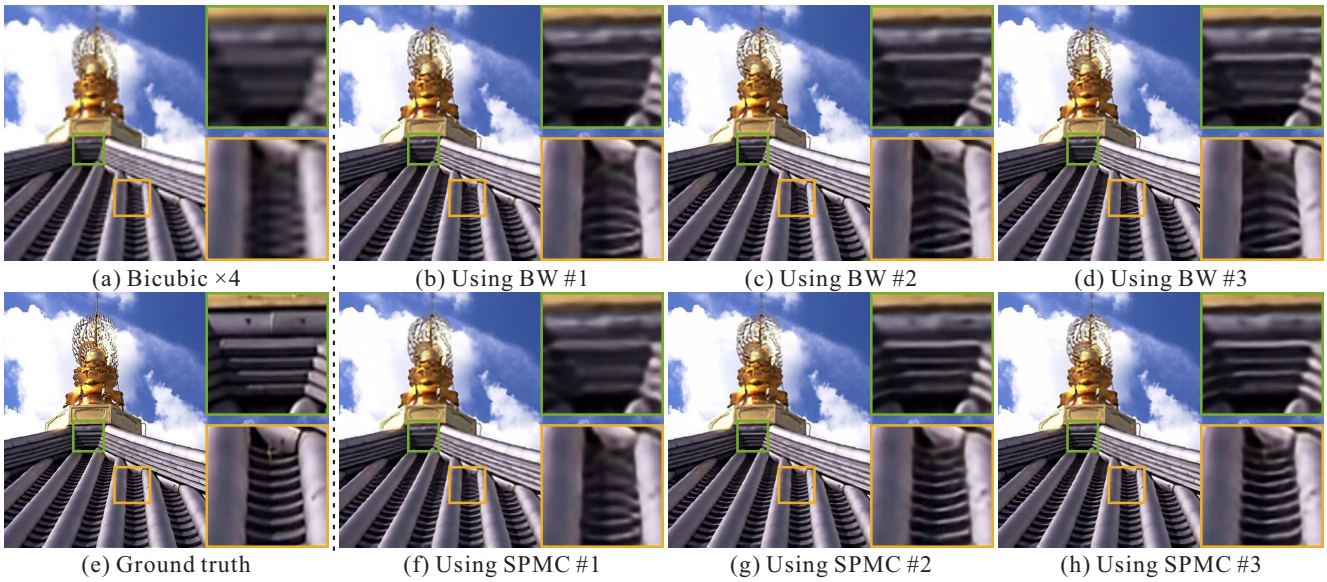


Figure 4. **Effectiveness of SPMC Layer** (F3-×4). (a) Bicubic ×4. (b)-(d) Output for each time step using **BW**. (e) Ground truth. (f)-(h) Outputs using **SPMC**.

Phase 2 We then fix the learned weights θ_{ME} and only train $\mathbf{Net}_{DF}$. This time we use Euclidean loss between our estimated HR reference frame and the ground truth as

$$\mathcal{L}_{SR} = \sum_{i=-T}^{T} \kappa_i \|I_0^H - I_0^{(i)}\|_2^2, \tag{13}$$

where $I_0^{(i)}$ is our network output in the ith time step, corresponding to reference frame I_0^L. $\{\kappa_i\}$ are the weights for each time step. We empirically set $\kappa_{-T} = 0.5$ and $\kappa_T = 1.0$, and linearly interpolate intermediate values.

Phase 3 In the last stage, we jointly tune the whole system using the total loss as

$$\mathcal{L} = \mathcal{L}_{SR} + \lambda_2 \mathcal{L}_{ME}, \tag{14}$$

where λ_2 is the weight balancing two losses.

4. Experiments

We conduct our experiments on a PC with an Intel Xeon E5 CPU and an NVIDIA Titan X GPU. We implement our framework on the TensorFlow platform [6], which enables us to easily develop our special layers and experiment with different network configurations.

Data Preparation For the super-resolution task, training data needs to be of high-quality without noise while containing rich fine details. To our knowledge, there is no such publicly available video dataset that is large enough to train our deep networks. We thus collect 975 sequences from high-quality 1080p HD video clips. Most of them are commercial videos shot with high-end cameras and contain both natural-world and urban scenes that have rich details. Each sequence contains 31 frames following the configuration of [19, 20, 23]. We downsample the original frames to 540×960 pixels as HR ground truth using bicubic interpolation. LR input is obtained by further downsampling HR frames to 270×480, 180×320 and 135×240 sizes. We randomly choose 945 of them as training data, and the rest 30 sequences are for validation and testing.

Model Training For model training, we use Adam solver [17] with learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We apply gradient clip only to weights of ConvLSTM module (clipped by global norm 3) to stabilize the training process. At each iteration, we randomly sample N_F consecutive frames (*e.g.* $N_F = 3, 5, 7$) from one sequence, and randomly crop a 100×100 image region as training input. The corresponding ground truth is accordingly cropped from the reference frame with size $100\alpha \times 100\alpha$ where α is the scaling factor. Above parameters are fixed for all experiments. Batch size varies according to different settings, which is determined as the maximal value allowed by GPU memory.

We first train the motion estimation module using only loss $\mathcal{L}_{ME}$ in Eq. (12) with $\lambda_1 = 0.01$. After about 70,000 iterations, we fix the parameters θ_{ME} and train the system using only loss $\mathcal{L}_{SR}$ in Eq. (13) for 20,000 iterations. Finally, all parameters are trained using total loss $\mathcal{L}$ in Eq. (14), λ_2 is empirically chosen as 0.01. All trainable variables are initialized using Xavier methods [9].

In the following analysis and experiments, we train sev-

Table 1. Performance of baseline models

Model (F3)	BW	DF-Bic	DF-0up	Ours
SPMCS ($\times 4$)	29.23 / 0.82	29.67 / 0.83	29.65 / 0.83	**29.69 / 0.84**

eral models under different settings. For simplicity, we use $\times(\cdot)$ to denote scaling factors (*e.g.* $\times 2$, $\times 3$, and $\times 4$). And F$(\cdot)$ is used as the number of input frames (*e.g.* F3, F5, and F7). Moreover, our ConvLSTM based DF net produces multiple outputs (one for each time step), we use $\{\#1, \#2, \cdots\}$ to index output.

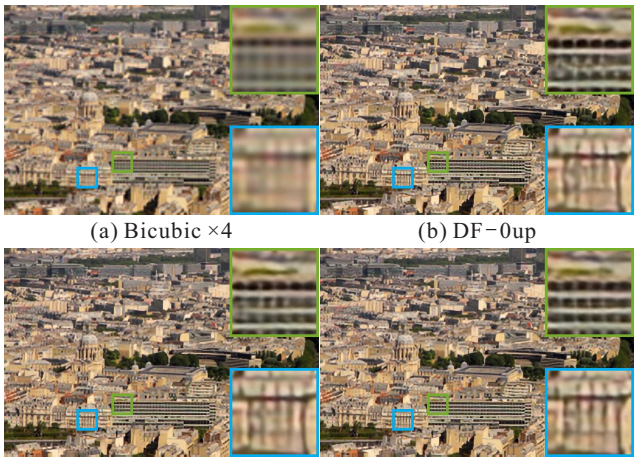
4.1. Effectiveness of SPMC Layer

We first evaluate the effectiveness of the proposed SPMC layer. For comparison, a baseline model **BW** (F3-$\times 4$) is used. It is achieved by fixing our system in Fig. 2, except replacing the SPMC layer with backward warping, followed by bicubic interpolation, which is a standard alignment procedure. An example is shown in Fig. 4. In Fig. 4(a), bicubic $\times 4$ for reference frame contains severe aliasing for the tile patterns. Baseline model **BW** produces 3 outputs corresponding to three time steps in Fig. 4(b)-(d). Although results are sharper when more frames are used, tile patterns are obviously wrong compared to ground truth in Fig. 4(e). This is due to loss of sub-pixel information as analyzed in Section 2. The results are similar to the output of single image SR, where the reference frame dominates.

As shown in Fig. 4(f), if we only use one input image in our method, the recovered pattern is also similar to Fig. 4(a)-(d). However, with more input frames fed into the system, the restored images dramatically improve, as shown in Fig. 4(g)-(h), which are both sharper and closer to the ground truth. Quantitative values on our validation set are listed in Table 1.

4.2. Detail Fusion vs. Synthesis

We further investigate if our recovered details truly exist in original frames. One example is already shown in Fig. 4. Here we conduct a more illustrative experiment by replacing all input frames with the same reference frame. Specifically, Fig. 5(f)-(h) are outputs using 3 consecutive frames (F3-$\times 3$). The numbers and logo are recovered nicely. However, if we only use 3 copies of the same reference frame as input and test them on the same pre-trained model, the results are almost the same as using only one frame. This manifests that our final result shown in Fig. 5(h) is truly recovered from the 3 different input frames based on their internal detail information, rather than synthesized from external examples because if the latter holds, the synthesized details should also appear even if we use only one reference frame.



(a) Bicubic $\times 4$ (b) DF$-$0up (c) DF$-$Bic (d) Ours

Figure 6. **Detail fusion net with various inputs.**

4.3. DF-Net with Various Inputs

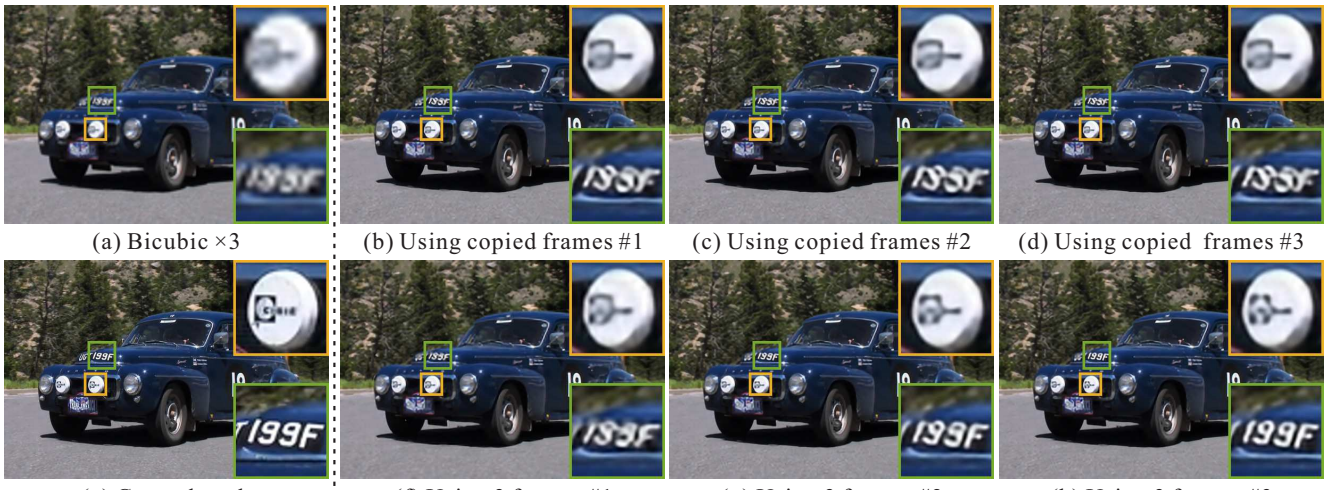
Our proposed detail fusion (DF) net takes only J_i^H as input. To further evaluate if the reference frame is needed, we design two baseline models. Model **DF-bic** and **DF-0up** respectively add bicubic and zero-upsampled I_0^L as another channel of input to DF net. Visual comparison in Fig. 6 shows that although all models can recover reasonable details, the emphasis on the reference frame may mislead detail recovery and slightly degrade results quantitatively on the evaluation set (see Table 1).

4.4. Comparisons with Video SR Methods

We compare our method with previous video SR ones on the evaluation dataset. BayesSR [20, 23] is an important method that iteratively estimates motion flow, blur kernel, noise and the HR image. DESR [19] ensembles "draft" based on estimated flow, which makes it an intermediate solution between traditional and CNN-based methods. We also include a recent deep-learning-based method VSRnet [14] in comparison. We use author-provided implementation for all these methods. VESPCN [2] did not provide code or pre-trained model, so we only list their reported PSNR/SSIM on the 4-video dataset **VID4** [20]. The quantitative results are listed in Table 2. Visual comparison is shown in Fig. 7.

4.5. Comparisons with Single Image SR

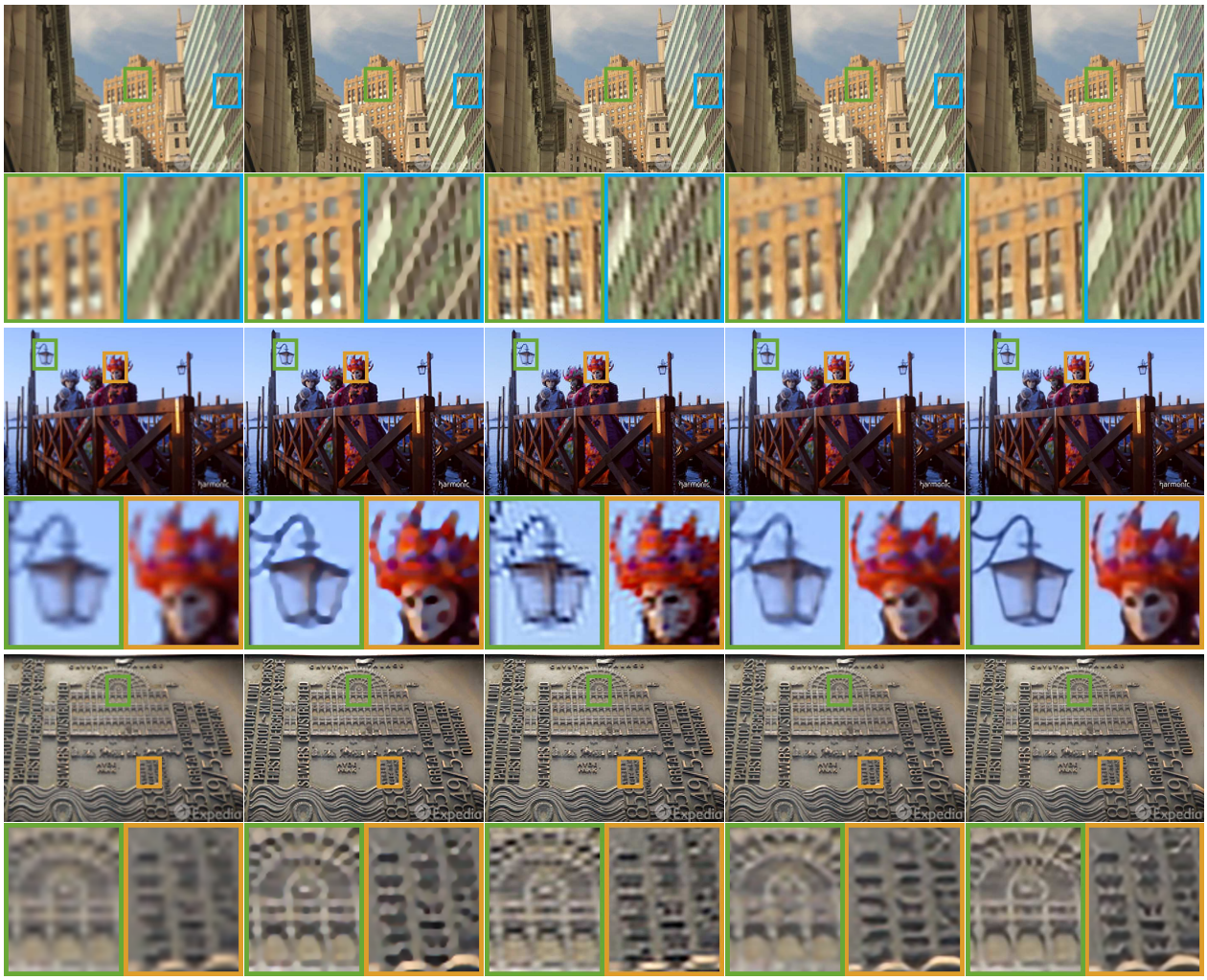
Since our framework is flexible, we set $N_F = 1$ to turn it into a single image SR solution. We compare this approach with three recent image SR methods: SRCNN [3], FSRCNN [4] and VDSR [15], on dataset **Set5** [1] and **Set14** [32]. To further compare the performance of using one and multiple frames, we also run all single-image SR methods and ours under F3 setting on our evaluation dataset **SPMCS**. The quantitative results are listed in Table 3.



(a) Bicubic ×3 (b) Using copied frames #1 (c) Using copied frames #2 (d) Using copied frames #3

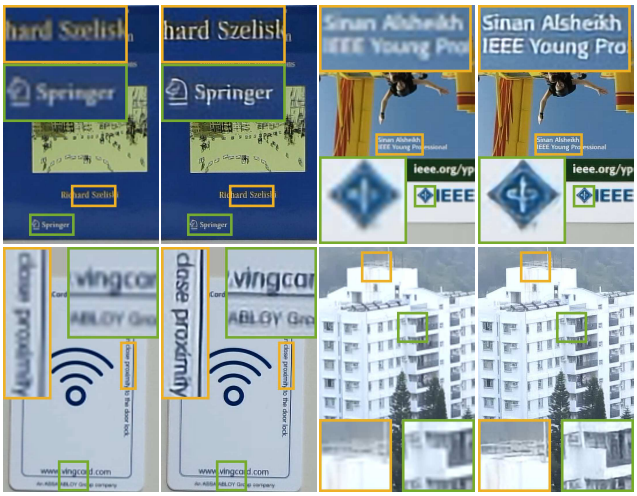
(e) Ground truth (f) Using 3 frames #1 (g) Using 3 frames #2 (h) Using 3 frames #3

Figure 5. **SR using multiple frames** (F3-×3). (a) Bicubic ×3. (b)-(d) Output for each time step using 3 reference frames that are with the same content. (e) Ground truth. (f)-(h) Output using 3 consecutive frames.



(a) Bicubic ×4 (b)BayesSR (c) DESR (d) VSRnet (e) Ours（F5−×4）

Figure 7. Comparisons with video SR methods.

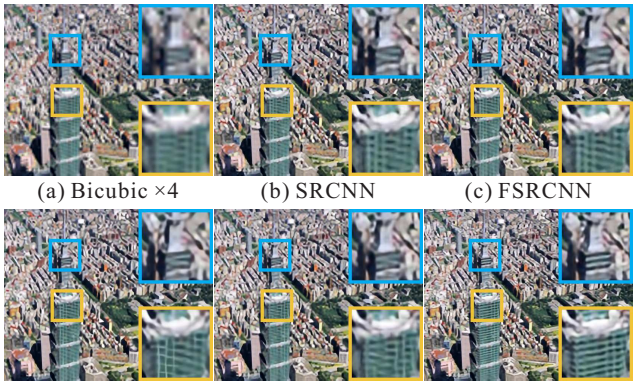


(a) Bicubic ×4 (b) Ours (c) Bicubic ×4 (d) Ours

Figure 8. **Real-world examples under configuration** (F7-×4).

Table 2. Comparison with video SR methods (PSNR/SSIM)

Method (F3)	Bicubic	BayesSR	DESR	VSRnet	Ours (F3)
SPMCS×2	32.48 / 0.92	31.85 / 0.92	-	33.39 / 0.94	**36.71 / 0.96**
SPMCS×3	28.85 / 0.82	29.42 / 0.87	-	28.55 / 0.85	**31.92 / 0.90**
SPMCS×4	27.02 / 0.75	27.87 / 0.80	26.64 / 0.76	24.76 / 0.77	**29.69 / 0.84**
Method (F5)	Bicubic	BayesSR	DESR	VSRNet	Ours (F5)
SPMCS×2	32.48 / 0.92	31.82 / 0.92	-	35.44 / 0.95	**36.62 / 0.96**
SPMCS×3	28.85 / 0.82	29.55 / 0.87	-	30.73 / 0.88	**32.10 / 0.90**
SPMCS×4	27.02 / 0.75	28.03 / 0.81	26.97 / 0.77	28.35 / 0.79	**29.89 / 0.84**
Method (F3)	BayesSR	DESR	VSRNet	VESPCN	Ours (F3)
Vid4×3	25.64 / 0.80	-	25.31 / 0.76	27.25 / **0.84**	**27.49 / 0.84**
Vid4×4	24.42 / 0.72	23.50 / 0.67	22.81 / 0.65	25.35 / **0.76**	**25.52 / 0.76**



(a) Bicubic ×4 (b) SRCNN (c) FSRCNN

(d) VDSR (e) Ours (F1) (f) Ours (F3)

Figure 9. **Comparisons with single image SR methods**. (a) Bicubic ×4. (b)-(d) Output from image SR methods. (e) Our result using 1 frame. (f) Our result using 3 frames.

For the F1 setting on **Set5** and **Set14**, our method produces comparable or slightly lower-PSNR or -SSIM results. Under the F3 setting, our method outperforms them by a large margin, indicating that our multi-frame setting can effectively fuse information in multiple frames. An example is shown in Fig. 9, where single image SR cannot recover the tiled structure of the building. In contrast, our F3 model can faithfully restore it.

Table 3. Comparison with image SR methods (PSNR/SSIM)

Method	SRCNN	FSRCNN	VDSR	Ours (F1)	Ours (F3)
Set 5 (×2)	36.66 / 0.95	37.00 / 0.96	**37.53 / 0.96**	37.35 / **0.96**	-
Set 5 (×3)	32.75 / 0.91	33.16 / 0.92	**33.66 / 0.92**	33.45 / **0.92**	-
Set 5 (×4)	30.49 / 0.86	30.71 / 0.88	**31.35 / 0.88**	30.96 / 0.87	-
Set 14 (×2)	32.45 / 0.91	32.63 / 0.91	**33.03 / 0.91**	32.70 / **0.91**	-
Set 14 (×3)	29.30 / 0.82	29.43 / 0.83	**29.77 / 0.83**	29.36 / **0.83**	-
Set 14 (×4)	27.45 / 0.75	27.59 / 0.77	**28.01 / 0.77**	27.57 / 0.76	-
SPMCS (×2)	35.20 / 0.95	35.56 / 0.95	36.14 / 0.96	36.23 / 0.96	**36.71 / 0.96**
SPMCS (×3)	30.66 / 0.87	30.87 / 0.88	31.26 / 0.89	31.18 / 0.88	**31.92 / 0.90**
SPMCS (×4)	28.29 / 0.79	28.43 / 0.79	28.80 / 0.81	28.80 / 0.80	**29.69 / 0.84**

4.6. Real-World Examples

The LR images in the above evaluation are produced though downsampling (bicubic interpolation). Although this is a standard approach for evaluation [3, 4, 14, 15, 19, 20], the generated LR images may not fully resemble the real-world cases. To verify the effectiveness of our method on real-world data, we captured four examples as shown in Fig. 8. For each object, we capture a short video using a hand-held cellphone camera, and extract 31 consecutive frames from it. We then crop a 135×240 region from the center frame, and use TLD tracking [12] to track and crop the same region from all other frames as the input data to our system. Fig. 8 shows the SR result of the center frame for each sequence. Our method faithfully recovers the textbook characters and fine image details using the F7-×4 model. More examples are included in our supplementary material.

4.7. Model Complexity and Running Time

Using our un-optimized TensorFlow code, the F7-×4 model takes about $0.26s$ to process 7 input images with size 180×120 for one HR output. In comparison, reported timings for other methods (F31) are 2 hours for Liu *et al*. [20], 10 min. for Ma *et al*. [23], and 8 min. for DESR [19]. VSRnet [14] requires ≈40s for F5 configuration. Our method is further accelerated to 0.19s for F5 and 0.14s for F3.

5. Concluding Remarks

We have proposed a new deep-learning-based approach for video SR. Our method includes a sub-pixel motion compensation layer that can better handle inter-frame motion for this task. Our detail fusion (DF) network that can effectively fuse image details from multiple images after SPMC alignment. We have conducted extensive experiments to validate the effectiveness of each module. Results show that our method can accomplish high-quality results both qualitatively and quantitatively, at the same time flexible regarding scaling factors and numbers of input frames.

References

[1] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. 2012. 6

[2] J. Caballero, C. Ledig, A. Aitken, A. Acosta, J. Totz, Z. Wang, and W. Shi. Real-time video super-resolution with spatio-temporal networks and motion compensation. *arXiv preprint arXiv:1611.05250*, 2016. 1, 2, 3, 6

[3] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199, 2014. 2, 6, 8

[4] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, pages 391–407. Springer, 2016. 2, 6, 8

[5] M. Elad and Y. Hel-Or. A fast super-resolution reconstruction algorithm for pure translational motion and common space-invariant blur. *IEEE Transactions on Image Processing*, 10(8):1187–1193, 2001. 2

[6] M. A. et. al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 5

[7] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar. Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, 13(10):1327–1344, 2004. 1, 2

[8] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *ICCV*, 2015. 2, 3

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010. 5

[10] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *NIPS*, pages 2017–2025, 2015. 2, 4

[11] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 2

[12] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(7):1409–1422, 2012. 8

[13] A. Kanazawa, D. W. Jacobs, and M. Chandraker. Warpnet: Weakly supervised matching for single-view reconstruction. *CVPR*, 2016. 2

[14] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE Transactions on Computational Imaging*, 2(2):109–122, 2016. 1, 2, 6, 8

[15] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, June 2016. 2, 6, 8

[16] J. Kim, J. Kwon Lee, and K. Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, June 2016. 2

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 5

[18] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016. 2

[19] R. Liao, X. Tao, R. Li, Z. Ma, and J. Jia. Video super-resolution via deep draft-ensemble learning. In *ICCV*, pages 531–539, 2015. 1, 2, 5, 6, 8

[20] C. Liu and D. Sun. A bayesian approach to adaptive video super resolution. In *CVPR*, pages 209–216. IEEE, 2011. 1, 2, 5, 6, 8

[21] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. 2017. 4

[22] W. Luo, A. G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *CVPR*, pages 5695–5703, 2016. 2

[23] Z. Ma, R. Liao, X. Tao, L. Xu, J. Jia, and E. Wu. Handling motion blur in multi-frame super-resolution. In *CVPR*, pages 5224–5232, 2015. 1, 2, 5, 6, 8

[24] X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *NIPS*, pages 2802–2810, 2016. 4

[25] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016. 2, 3

[26] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, pages 1874–1883, 2016. 1, 2

[27] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang. Deep video deblurring. *arXiv preprint arXiv:1611.08387*, 2016. 4

[28] H. Takeda, P. Milanfar, M. Protter, and M. Elad. Super-resolution without explicit subpixel motion estimation. *IEEE Transactions on Image Processing*, 18(9):1958–1975, 2009. 1

[29] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, pages 802–810, 2015. 1, 4

[30] J. J. Yu, A. W. Harley, and K. G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. *arXiv preprint arXiv:1608.05842*, 2016. 2, 3, 4

[31] J. Zbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17:1–32, 2016. 2

[32] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *International Conference on Curves and Surfaces*, pages 711–730. Springer, 2010. 6