Supplementary Materials of Towards Interpretable Video Super-Resolution via Alternating Optimization

Jiezhang Cao[1], Jingyun Liang[1], Kai Zhang[1⋆], Wenguan Wang[1], Qin Wang[1], Yulun Zhang[1], Hao Tang[1], and Luc Van Gool[1,2]

[1]Computer Vision Lab, ETH Zürich, Switzerland [2]KU Leuven, Belgium
{jiezhang.cao, jingyun.liang, kai.zhang, wenguan.wang, qin.wang, yulun.zhang, hao.tang, vangool}@vision.ee.ethz.ch
https://github.com/caojiezhang/DAVSR

In the supplementary materials, we first provide the theoretical analysis of the method in Section A. In Section B, we provide more details of RVE. In Section C, we introduce more experiment settings. In Section D, we provide more visual comparisons with state-of-the-art methods. In Section E, we provide the comparisons of the model size, reference time and performance. Last, we provide further analyses on a blurry kernel for real videos, an impact of the order of video restoration and an impact of number of frames in Section F.

A Theoretical Analysis

Based on the linearity of convolution and the properties of downsampling, there exist a blurring matrix $\boldsymbol{H}$ and a decimation matrix $\boldsymbol{S}$, then we can rewrite the video degradation model in the paper as follows:

$$\boldsymbol{Y} = (\boldsymbol{X} \otimes \boldsymbol{K}) \downarrow_{\boldsymbol{s}} + \boldsymbol{N} \tag{S1}$$

$$\Rightarrow \quad \boldsymbol{y} = \boldsymbol{S}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{S2}$$

where $\boldsymbol{y} = \mathrm{vec}(\boldsymbol{Y}), \boldsymbol{x} = \mathrm{vec}(\boldsymbol{X})$ and $\boldsymbol{n} = \mathrm{vec}(\boldsymbol{N})$. Here, $\mathrm{vec}(\cdot)$ is the vectorization of a tensor, *i.e.*, stacking the corresponding frames into column vectors in the order which is the same as "view" in Pytorch.

To develop our analysis, we use a block circulant matrix circulant blocks assumption of the blurring matrix, which widely used in many studies [9,11,5].

Assumption 1 *[15] The blurring matrix $\boldsymbol{H}$ is the matrix representation of the cyclic convolution, i.e., $\boldsymbol{H}$ is a block circulant matrix with circulant blocks.*

In this definition, the convolution is periodic because of the shift-invariant blurring kernel and the boundary conditions. Note that the assumption can be used in any kind of blurring, *e.g.*, motion blur, out-of-focus blur and atmospheric turbulence, *etc.* Based on the assumption of cyclic convolution, the blurring matrix and its conjugate transpose can be decomposed as

$$\boldsymbol{H} = \boldsymbol{F}^* \boldsymbol{\Lambda} \boldsymbol{F} \quad \text{and} \quad \boldsymbol{H}^* = \boldsymbol{F}^* \boldsymbol{\Lambda}^* \boldsymbol{F}, \tag{S3}$$

⋆ Corresponding author.

where the matrices $\boldsymbol{F}$ and $\boldsymbol{F}^*$ are the Fourier and inverse Fourier transforms, which satisfy $\boldsymbol{F}\boldsymbol{F}^* = \boldsymbol{F}^*\boldsymbol{F} = \boldsymbol{I}$. The matrix $\boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{F}\boldsymbol{h})$ is a diagonal, and its diagonal elements are the Fourier coefficients of the first column of the blurring matrix $\boldsymbol{H}$, denoted as $\boldsymbol{h}$.

In addition, we also use an assumption of the decimation matrix for down-sampling, which is widely used in many literatures [12,13,9].

Assumption 2 *[15] The decimation matrix $\boldsymbol{S}$ is a down-sampling operator, while its conjugate transpose $\boldsymbol{S}$ interpolates the decimated video with zeros.*

In this assumption, the decimation matrix satisfies $\boldsymbol{S}\boldsymbol{S}^* = \boldsymbol{I}$.

We can rewrite the space-time video super-resolution problem as:

$$\min_{\boldsymbol{X}} E(\boldsymbol{X}) := \frac{1}{2\sigma^2}\|\boldsymbol{Y} - (\boldsymbol{X} \otimes \boldsymbol{K}) \downarrow_{\boldsymbol{s}} \|^2 + \lambda\Phi(\boldsymbol{X}), \tag{S4}$$

$$\Rightarrow \quad \min_{\boldsymbol{x}} E(\boldsymbol{x}) := \frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{S}\boldsymbol{H}\boldsymbol{x}\|^2 + \lambda\Phi(\boldsymbol{x}). \tag{S5}$$

Based on the Half-Quadratic Splitting (HQS) algorithm [1,14], we introduce an auxiliary variable $\boldsymbol{z}$ and a penalty parameter μ to reformulate Eq. (S5) as

$$E(\boldsymbol{x}, \boldsymbol{z}) = \frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{S}\boldsymbol{H}\boldsymbol{x}\|^2 + \lambda\Phi(\boldsymbol{x}) + \frac{\mu}{2}\|\boldsymbol{z} - \boldsymbol{x}\|^2. \tag{S6}$$

Then, Eq. (S6) can be solved by alternately optimizing two sub-problems Eq. (S7) (for $\boldsymbol{z}$) and Eq. (S8) (for $\boldsymbol{x}$) as follows

$$\begin{cases} \boldsymbol{z}_k = \arg\min_{\boldsymbol{z}} \|\boldsymbol{y} - \boldsymbol{S}\boldsymbol{H}\boldsymbol{z}\|^2 + \mu\sigma^2\|\boldsymbol{z} - \boldsymbol{x}_{k-1}\|^2, & \text{(S7)} \\ \boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \frac{\mu}{2}\|\boldsymbol{z}_k - \boldsymbol{x}\|^2 + \lambda\Phi(\boldsymbol{x}). & \text{(S8)} \end{cases}$$

One can directly calculate the close-form solution of the problem (S7) as

$$\boldsymbol{z}_k = \left(\boldsymbol{H}^*\boldsymbol{S}^*\boldsymbol{S}\boldsymbol{H} + \mu\sigma^2\boldsymbol{I}\right)^{-1}\left(\boldsymbol{H}^*\boldsymbol{S}^*\boldsymbol{y} + \mu\sigma^2\boldsymbol{x}_{k-1}\right). \tag{S9}$$

However, the solution Eq. (S9) requires the inversion of a high dimensional matrix, whose computational complexity is $O(T_h^3 W_h^3 H_h^3)$.

Based on [15], we extend the case of 2D to 3D, and the decimation matrix $\boldsymbol{S}$ has the following property in the Fourier Domain,

Lemma 1. *[10] Let $\boldsymbol{J}_s \in \mathbb{R}^{d\times d}$ be a matrix of ones, then the decimation matrix has the following property:*

$$\boldsymbol{F}\boldsymbol{S}^*\boldsymbol{S}\boldsymbol{F}^* = \frac{1}{s_w s_h s_t}(\boldsymbol{J}_{s_t} \circledast \boldsymbol{I}_t) \circledast (\boldsymbol{J}_{s_w} \circledast \boldsymbol{I}_w) \circledast (\boldsymbol{J}_{s_h} \circledast \boldsymbol{I}_h), \tag{S10}$$

where $\boldsymbol{I}_t \in \mathbb{R}^{t\times t}$ is the an identiry matrix and $\circledast$ is the Kronecker product.

Lemma 2. *Given a Fourier matrix $\boldsymbol{F}$ and a decimation matrix $\boldsymbol{S}$, we have*

$$\boldsymbol{\Lambda}^*\boldsymbol{F}\boldsymbol{S}^*\boldsymbol{S}^*\boldsymbol{F}^*\boldsymbol{\Lambda} = \frac{1}{s}[\boldsymbol{\Lambda}_1, ..., \boldsymbol{\Lambda}_s]^\top[\boldsymbol{\Lambda}_1, ..., \boldsymbol{\Lambda}_s] := \frac{1}{s}\underline{\boldsymbol{\Lambda}}^*\underline{\boldsymbol{\Lambda}}. \tag{S11}$$

Proof. Based on Lemma (1) and the property of the Kronecker product, we have

$$
\begin{aligned}
&\boldsymbol{\Lambda}^*\boldsymbol{F}\boldsymbol{S}^*\boldsymbol{S}^*\boldsymbol{F}^*\boldsymbol{\Lambda}\\
=&\frac{1}{s}\boldsymbol{\Lambda}^*(\boldsymbol{J}_{s_t}\circledast\boldsymbol{I}_t)\circledast(\boldsymbol{J}_{s_w}\circledast\boldsymbol{I}_w)\circledast(\boldsymbol{J}_{s_h}\circledast\boldsymbol{I}_h)\boldsymbol{\Lambda}\\
=&\frac{1}{s}\boldsymbol{\Lambda}^*((\mathbf{1}_{s_t}\mathbf{1}_{s_t}^\top)\circledast(\boldsymbol{I}_t\boldsymbol{I}_t))\circledast((\mathbf{1}_{s_w}\mathbf{1}_{s_w}^\top)\circledast(\boldsymbol{I}_w\boldsymbol{I}_w))\circledast((\mathbf{1}_{s_h}\mathbf{1}_{s_h}^\top)\circledast(\boldsymbol{I}_h\boldsymbol{I}_h))\boldsymbol{\Lambda}\\
=&\frac{1}{s}\boldsymbol{\Lambda}^*((\mathbf{1}_{s_t}\circledast\boldsymbol{I}_t)(\mathbf{1}_{s_t}^\top\circledast\boldsymbol{I}_t))\circledast((\mathbf{1}_{s_w}\circledast\boldsymbol{I}_w)(\mathbf{1}_{s_w}^\top\circledast\boldsymbol{I}_w))\circledast((\mathbf{1}_{s_h}\circledast\boldsymbol{I}_h)(\mathbf{1}_{s_h}^\top\circledast\boldsymbol{I}_h))\boldsymbol{\Lambda}\\
=&\frac{1}{s}\boldsymbol{\Lambda}^*\big([\boldsymbol{I}_t,...,\boldsymbol{I}_t]^\top[\boldsymbol{I}_t,...,\boldsymbol{I}_t]\big)\circledast\big([\boldsymbol{I}_w,...,\boldsymbol{I}_w]^\top[\boldsymbol{I}_w,...,\boldsymbol{I}_w]\big)\circledast\big([\boldsymbol{I}_h,...,\boldsymbol{I}_h]^\top[\boldsymbol{I}_h,...,\boldsymbol{I}_h]\big)\,\boldsymbol{\Lambda}\\
=&\frac{1}{s}\boldsymbol{\Lambda}^*\big([\boldsymbol{I}_t,...,\boldsymbol{I}_t]^\top\circledast[\boldsymbol{I}_w,...,\boldsymbol{I}_w]^\top\circledast[\boldsymbol{I}_h,...,\boldsymbol{I}_h]^\top\big)\,([\boldsymbol{I}_t,...,\boldsymbol{I}_t]\circledast[\boldsymbol{I}_w,...,\boldsymbol{I}_w]\circledast[\boldsymbol{I}_h,...,\boldsymbol{I}_h])\boldsymbol{\Lambda}\\
=&\frac{1}{s}\boldsymbol{\Lambda}^*[\boldsymbol{I}_{twh},...,\boldsymbol{I}_{twh}]^\top[\boldsymbol{I}_{twh},...,\boldsymbol{I}_{twh}]\boldsymbol{\Lambda}\\
=&\frac{1}{s}[\boldsymbol{\Lambda}_1,...,\boldsymbol{\Lambda}_s]^\top[\boldsymbol{\Lambda}_1,...,\boldsymbol{\Lambda}_s]\\
=&\frac{1}{s}\underline{\boldsymbol{\Lambda}}^*\underline{\boldsymbol{\Lambda}}.
\end{aligned}
$$

Lemma 3. ***(Woodbury formula** [3]) Given matrices $\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3$ and $\boldsymbol{A}_4$, the following equality holds conditional on the existence of $\boldsymbol{A}_1^{-1}$ and $\boldsymbol{A}_3^{-1}$, we have*

$$
(\boldsymbol{A}_1+\boldsymbol{A}_2\boldsymbol{A}_3\boldsymbol{A}_4)^{-1}=\boldsymbol{A}_1^{-1}-\boldsymbol{A}_1^{-1}\boldsymbol{A}_2(\boldsymbol{A}_3^{-1}+\boldsymbol{A}_4\boldsymbol{A}_1^{-1}\boldsymbol{A}_2)^{-1}\boldsymbol{A}_4\boldsymbol{A}_1^{-1}. \qquad \text{(S12)}
$$

Theorem 1 *Assume the blurring matrix is a block circulant matrix with circulant blocks, and the conjugate transpose of the decimation matrix interpolates the decimated image with zeros [15]. Then the solution of Eq. (S7) can be computed using the following closed-form expression*

$$
\boldsymbol{z}_k=\frac{1}{\mu\sigma^2}\boldsymbol{r}_{k-1}-\frac{1}{\mu\sigma^2}\boldsymbol{F}^*\underline{\boldsymbol{\Lambda}}^*\left(\mu\sigma^2 s\boldsymbol{I}+\underline{\boldsymbol{\Lambda}}\underline{\boldsymbol{\Lambda}}^*\right)^{-1}\underline{\boldsymbol{\Lambda}}\boldsymbol{F}\boldsymbol{r}_{k-1}, \qquad \text{(S13)}
$$

where $s=s_w s_h s_t$, $\boldsymbol{r}_{k-1}=\big(\boldsymbol{H}^*\boldsymbol{S}^*\boldsymbol{y}+\mu\sigma^2\boldsymbol{x}_{k-1}\big)$.

Proof. Based on Lemmas 1, 2 and 3, we have

$$
\begin{aligned}
\boldsymbol{z}_k=&\big(\boldsymbol{F}^*\boldsymbol{\Lambda}^*\boldsymbol{F}\boldsymbol{S}^*\boldsymbol{S}^*\boldsymbol{F}^*\boldsymbol{\Lambda}\boldsymbol{F}+\mu\sigma^2\boldsymbol{I}\big)^{-1}\boldsymbol{F}\left(\boldsymbol{H}^*\boldsymbol{S}^*\boldsymbol{y}+\mu\sigma^2\boldsymbol{x}_{k-1}\right) && \text{(S14)}\\
=&\boldsymbol{F}^*\left(\frac{1}{s}\underline{\boldsymbol{\Lambda}}^*\underline{\boldsymbol{\Lambda}}+\mu\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{F}\left(\boldsymbol{H}^*\boldsymbol{S}^*\boldsymbol{y}+\mu\sigma^2\boldsymbol{x}_{k-1}\right) && \text{(S15)}\\
=&\boldsymbol{F}^*\left(\frac{1}{s}\underline{\boldsymbol{\Lambda}}^*\underline{\boldsymbol{\Lambda}}+\mu\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{F}\boldsymbol{r}_{k-1} && \text{(S16)}\\
=&\boldsymbol{F}^*\left(\frac{1}{\mu\sigma^2}\boldsymbol{I}-\frac{1}{\mu\sigma^2}\underline{\boldsymbol{\Lambda}}^*\left(s\boldsymbol{I}+\frac{1}{\mu\sigma^2}\underline{\boldsymbol{\Lambda}}\underline{\boldsymbol{\Lambda}}^*\right)^{-1}\underline{\boldsymbol{\Lambda}}\frac{1}{\mu\sigma^2}\right)\boldsymbol{F}\boldsymbol{r}_{k-1} && \text{(S17)}\\
=&\frac{1}{\mu\sigma^2}\boldsymbol{r}_{k-1}-\frac{1}{\mu\sigma^2}\boldsymbol{F}^*\underline{\boldsymbol{\Lambda}}^*\left(\mu\sigma^2 s\boldsymbol{I}+\underline{\boldsymbol{\Lambda}}\underline{\boldsymbol{\Lambda}}^*\right)^{-1}\underline{\boldsymbol{\Lambda}}\boldsymbol{F}\boldsymbol{r}_{k-1}, && \text{(S18)}
\end{aligned}
$$

Eq. (S14) can be directly obtained by taking the derivative of Problem (S7) and make it equal to 0, *i.e.*,

$$\left(\boldsymbol{F}^*\boldsymbol{\Lambda}^*\boldsymbol{F}\boldsymbol{S}^*\boldsymbol{S}^*\boldsymbol{F}^*\boldsymbol{\Lambda}\boldsymbol{F}+\mu\sigma^2\boldsymbol{I}\right)\boldsymbol{z}_k-\left(\boldsymbol{H}^*\boldsymbol{S}^*\boldsymbol{y}+\mu\sigma^2\boldsymbol{x}_{k-1}\right)=0 \tag{S19}$$

$$\Rightarrow \quad \boldsymbol{z}_k=\boldsymbol{F}^*\left(\boldsymbol{\Lambda}^*\boldsymbol{F}\boldsymbol{S}^*\boldsymbol{S}^*\boldsymbol{F}^*\boldsymbol{\Lambda}+\mu\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{F}\left(\boldsymbol{H}^*\boldsymbol{S}^*\boldsymbol{y}+\mu\sigma^2\boldsymbol{x}_{k-1}\right). \tag{S20}$$

Eq. (S15) follows by Lemma 2, *i.e.*,

$$\boldsymbol{\Lambda}^*\boldsymbol{F}\boldsymbol{S}^*\boldsymbol{S}^*\boldsymbol{F}^*\boldsymbol{\Lambda}=\frac{1}{s}\underline{\boldsymbol{\Lambda}}^*\underline{\boldsymbol{\Lambda}}. \tag{S21}$$

Eq. (S16) can be rewritten by using

$$\boldsymbol{r}_{k-1}=\left(\boldsymbol{H}^*\boldsymbol{S}^*\boldsymbol{y}+\mu\sigma^2\boldsymbol{x}_{k-1}\right). \tag{S22}$$

Eq. (S17) can be calculated by using the Woodbury formula in Lemma 3.

Based on Theorem 1, we rewrite the blurring kernel and decimation matrix as original matrices, and the closed-form solution can be rewritten as the following expression in Theorem 2.

Theorem 2 *Let $\mathcal{F}$ and $\mathcal{F}^{-1}$ be the fast Fourier transform (FFT) and inverse FFT, and $\overline{\mathcal{F}}$ be the complex conjugate of $\mathcal{F}$. Assume the blur kernel $\boldsymbol{K}$ and the downsampling $\downarrow_s$ satisfy some properties [15]. Given a video $\boldsymbol{X}_{k-1}$ at the k-th iteration and a low-resolution video $\boldsymbol{Y}$, the solution of Eq. (S7) can be computed using the following closed-form expression, i.e.,*

$$\boldsymbol{Z}_k=\mathcal{F}^{-1}\left(\frac{1}{\alpha_k}\left(\mathcal{F}(\boldsymbol{R}_{k-1})-\overline{\mathcal{F}(\boldsymbol{K})}\left(\frac{\left(\mathcal{F}(\boldsymbol{K})\mathcal{F}(\boldsymbol{R}_{k-1})\right)\downarrow_s^a}{\left(s\alpha_k\boldsymbol{I}+\left(\mathcal{F}(\boldsymbol{K})\overline{\mathcal{F}(\boldsymbol{K})}\right)\downarrow_s^a\right)}\right)\uparrow_s^r\right)\right), \tag{S23}$$

where $\boldsymbol{R}_{k-1}=\overline{\mathcal{F}(\boldsymbol{K})}\mathcal{F}(\boldsymbol{Y}\uparrow_s)-\alpha_k\boldsymbol{X}_{k-1}$ with $\alpha_k=\mu_k\sigma^2$, $\uparrow_s$ is a standard s-fold upsampler , i.e., upsampling the spatial size by filling the new entries with zeros, $\uparrow_s^r$ is an upsampler by repeating the tensor the desired dimension, and $\downarrow_s^a$ is a distinct block downsampler, i.e., averaging the $s_t{\times}s_h{\times}s_w$ distinct blocks.

In Theorem 2, Eq. (S23) requires three FFT computations and one inverse FFT computation, which are the most expensive parts in implementation. Considering the computation complexities of FFT and inverse FFT, and the size of a frame is $T_hW_hH_h$ (here, we ignore the constant number of channels), the computation complexity of Eq. (S23) is $\mathcal{O}(T_hW_hH_h\log(T_hW_hH_h))$ [10], which is much smaller than the complexity of Eq. (S9) (*i.e.*, $\mathcal{O}(T_h^3W_h^3H_h^3)$) and can be computed efficiently on the modern GPU devices.

B More Details of RVE

We implement $\mathcal{G}$ in the RVE module by using the architecture of the flow-guided deformable alignment of [2] to predict offset and mask in DCN [16]. Given features $\boldsymbol{g}_k^i$, we use the optical-flow-guided deformable alignment as $\mathcal{G}$ to compute the features at every branch, *i.e.*,

$$\begin{aligned}\widetilde{\boldsymbol{z}}_k^i &= \mathcal{G}\left(\boldsymbol{g}_k^i, \widehat{\boldsymbol{z}}_k^{i-1}, \widehat{\boldsymbol{z}}_k^{i-2}, \boldsymbol{f}_k^{i\to i-1}, \boldsymbol{f}_k^{i\to i-2}\right), &\text{(S24)}\\ &= DCN([\widehat{\boldsymbol{z}}_k^{i-1}; \widehat{\boldsymbol{z}}_k^{i-2}], [\boldsymbol{o}_k^{i\to i-1}; \boldsymbol{o}_k^{i\to i-2}], [\boldsymbol{m}_k^{i\to i-1}; \boldsymbol{m}_k^{i\to i-2}]), &\text{(S25)}\end{aligned}$$

where $[\cdot;\cdot]$ is the concatenation along channel dimension, the offsets $\boldsymbol{o}_k^{i\to i-1}$ and $\boldsymbol{o}_k^{i\to i-2}$ and masks $\boldsymbol{m}_k^{i\to i-1}$ and $\boldsymbol{m}_k^{i\to i-2}$ are formulated as

$$\begin{aligned}\boldsymbol{o}_k^{i\to i-p} &= \boldsymbol{f}_k^{i\to i-p} + \text{Conv}([\boldsymbol{g}_k^i; \bar{\boldsymbol{z}}_k^{i-1}; \bar{\boldsymbol{z}}_k^{i-2}]), &\text{(S26)}\\ \boldsymbol{m}_k^{i\to i-p} &= \text{Sigmoid}(\text{Conv}([\boldsymbol{g}_k^i; \bar{\boldsymbol{z}}_k^{i-1}; \bar{\boldsymbol{z}}_k^{i-2}])), &\text{(S27)}\end{aligned}$$

where $p = 1, 2$, and $\bar{\boldsymbol{z}}_k^{i-1}$ and $\bar{\boldsymbol{z}}_k^{i-2}$ are warped features using the optical flows $\boldsymbol{f}_k^{i\to i-1}$ and $\boldsymbol{f}_k^{i\to i-2}$, *i.e.*,

$$\begin{aligned}\bar{\boldsymbol{z}}_k^{i-1} &= \text{warp}(\widehat{\boldsymbol{z}}_k^{i-1}, \boldsymbol{f}_k^{i\to i-1}), &\text{(S28)}\\ \bar{\boldsymbol{z}}_k^{i-2} &= \text{warp}(\widehat{\boldsymbol{z}}_k^{i-2}, \boldsymbol{f}_k^{i\to i-2}), &\text{(S29)}\end{aligned}$$

where warp$(\cdot)$ is a warp function according to the optical flow.

C More Experiment Settings

For the flow network, we use the pre-trained SpyNet [7], and set the initial learning rate to 2.5×10^{-5}. In the FAT layer, we use torch.rfft and torch.irfft to calculate FFT and inverse FFT operators, respectively. At the beginning, μ_k is initialized as a small value to accelerate the convergence and then increased gradually during optimization. In the RVE layer, β_k decreases as the iteration increases, whose setting is the same as [14]. At the iteration k=0, we initialize $\boldsymbol{X}_0$ by interpolating $\boldsymbol{Y}$ with scale factor $\boldsymbol{s}$ via the Bicubic and some interpolation approach (*e.g.*, linear interpolation or SuperSloMo [4]) in space and time. The number of the input LSTR frames is 5, and the number of the output HSTR frames is 25.

D More Qualitative Comparisons

We provide more visual results of different models on REDS4 in Fig. A. These results demonstrate that our model is able to synthesize frames with better quality than two-stage and three-stage methods. We also show an example of a video (clip_011) of our model in the attachment.

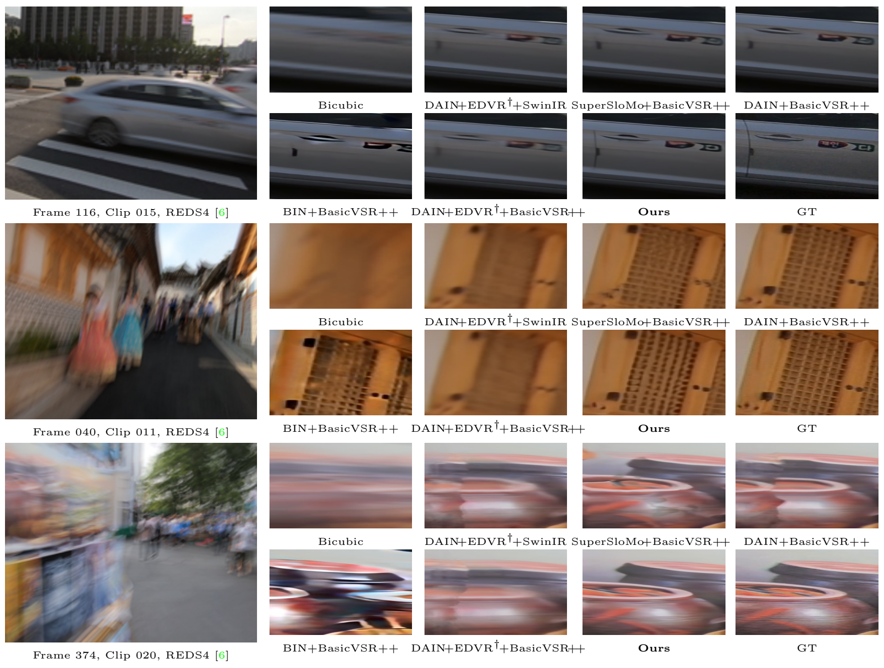

Fig. A: Visual comparison of different methods on **REDS4**.

E More Discussions on Model size and Inference Time

We compare the model size and inference time of different models on the REDS4 dataset in Table A. Our model achieves the smallest model size, the fastest reference time and the largest PSNR/SSIM. For example, it is more than 3× smaller than SuperSloMo+BasicVSR++* and it is also more than 2× faster than SuperSloMo+EDVR†+BasicVSR++. These results demonstrate the superiority of our one-stage framework over two-stage and three-stage frameworks.

Table A: Comparison of model size, inference time and performance on REDS4. The best results are **highlighted**.

Methods	Model size (M)	Inference time (s)	PSNR/SSIM
BIN+EDVR	25.3	0.658	23.52/0.662
BIN+BasicVSR	11.0	0.343	24.06/0.678
BIN+IconVSR	13.4	0.350	24.24/0.687
BIN+BasicVSR++	12.0	0.357	24.44/0.711
CDVD-TSP+Zooming Slow-Mo	27.3	3.460	25.99/0.759
EDVR+Zooming Slow-Mo	31.7	0.438	26.12/0.764
DAIN+CDVD-TSP+SwinIR	52.1	-	25.01/0.720
DAIN+CDVD-TSP+EDVR	60.8	4.104	25.38/0.737
DAIN+CDVD-TSP+BasicVSR	46.5	3.788	25.61/0.748
DAIN+CDVD-TSP+IconVSR	48.9	3.796	25.74/0.753
DAIN+CDVD-TSP+BasicVSR++	47.5	3.803	25.90/0.758
DAIN+EDVR†+SwinIR	56.5	-	25.43/0.730
DAIN+EDVR†+EDVR	65.2	1.082	25.78/0.747
DAIN+EDVR†+BasicVSR	50.9	0.767	25.81/0.748
DAIN+EDVR†+IconVSR	53.3	0.774	25.89/0.752
DAIN+EDVR†+BasicVSR++	51.9	0.781	25.98/0.755
DAIN+EDVR*	44.6	0.704	25.77/0.747
DAIN+BasicVSR*	30.3	0.389	26.08/0.763
DAIN+IconVSR*	32.7	0.396	26.24/0.768
DAIN+BasicVSR++*	31.3	0.403	26.53/0.778
SuperSloMo+CDVD-TSP+SwinIR	47.9	-	25.26/0.726
SuperSloMo+CDVD-TSP+EDVR	56.6	4.058	25.62/0.739
SuperSloMo+CDVD-TSP+BasicVSR	42.3	3.743	25.88/0.752
SuperSloMo+CDVD-TSP+IconVSR	44.7	3.750	26.04/0.756
SuperSloMo+CDVD-TSP+BasicVSR++	43.3	3.757	26.24/0.762
SuperSloMo+EDVR†+SwinIR	52.3	-	25.60/0.732
SuperSloMo+EDVR†+EDVR	61.0	1.036	26.05/0.751
SuperSloMo+EDVR†+BasicVSR	46.7	0.721	26.06/0.753
SuperSloMo+EDVR†+IconVSR	49.1	0.728	26.16/0.756
SuperSloMo+EDVR†+BasicVSR++	47.7	0.735	26.26/0.759
SuperSloMo+EDVR*	40.4	0.658	26.05/0.752
SuperSloMo+BasicVSR*	26.1	0.343	26.41/0.769
SuperSloMo+IconVSR*	82.5	0.350	26.60/0.775
SuperSloMo+BasicVSR++*	27.1	0.357	26.93/0.785
Ours	**8.0**	**0.29**	**29.12/0.859**

F Further Analysis

F.1 Blurry Kernel for Real Video

Blur videos can be generated by two ways: 1) convolving with a blur kernel and 2) averaging several successive frames. Our chosen kernel assumption is based on the latter which has been pointed out to be more realistic for its nonuniformity properties [31]. Note that the commonly used blur dataset (*i.e.*, REDS [31]) also adopts such assumption. To demonstrate the effectiveness for real applications, we show in Fig. B that our trained deep model can generalize well to real videos [8]. Without knowing the underlying kernels, our model is able to remove the bluriness, and outperforms the baseline (DAIN+EDVR+BasicVSR++).

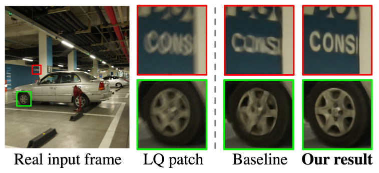

Fig. B: Visual comparison on RealBlur for real application.

The generalization performance on real-world videos can be further improved if more types of kernels are considered. To this end, we additionally train our model by adding Gaussian/disk blur to simulate camera-specific defocus. Then we test our model on synthetic data of REDS4 with the above blur. As shown in Table B, the model trained with these two blur types achieves better performance.

Table B: Comparison of other kernels on REDS4.

Methods	PSNR
w/ dynamic blur	28.78
w/ dynamic and Gaussian/disk blur	28.92

F.2 Impact of the Order of Video Restoration

To investigate the impact of the order of VFI, VD and VSR in baseline models, we change the order for SuperSloMo, EDVR and BasicVSR++ and provide the results in Table C. Since "VFI→VD→VSR" has slightly better performance than other orders, we choose it for the comparison in the paper.

Table C: Comparison of different order of VFI, VD and VSR.

Order	PSNR/SSIM
VFI→VD→VSR	26.26/0.759
VD→VFI→VSR	26.24/0.759
VSR→VFI→VD	26.23/0.756
Ours	**29.12/0.859**

F.3 Impact of Number of Frames

Using more frames can reduce the approximation error of FFT in the analytic solution and benefit from information accumulation in the recurrent video enhancement layer. As shown in Table D, more training and testing frames (denoted as n_{train} and n_{test}) lead to better PSNR/SSIM results. To balance the performance and computation cost, we trained our model with 5 blurry LR frames and 25 clean HR frames.

Table D: PSNR/SSIM of different n_{train} and n_{test}.

#frames	n_{test}=5	n_{test}=10
n_{train}=4	28.40/0.848	29.02/0.852
n_{train}=5	28.81/0.857	29.12/0.859
n_{train}=6	28.89/0.860	29.17/0.862

References

1. Afonso, M.V., Bioucas-Dias, J.M., Figueiredo, M.A.: Fast image recovery using variable splitting and constrained optimization. IEEE Transactions on Image Processing **19**(9), 2345–2356 (2010)
2. Chan, K.C., Zhou, S., Xu, X., Loy, C.C.: Basicvsr++: Improving video super-resolution with enhanced propagation and alignment. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 5972–5981 (2022)
3. Hager, W.W.: Updating the inverse of a matrix. SIAM review **31**(2), 221–239 (1989)
4. Jiang, H., Sun, D., Jampani, V., Yang, M.H., Learned-Miller, E., Kautz, J.: Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 9000–9008 (2018)
5. Lin, Z., Shum, H.Y.: Fundamental limits of reconstruction-based superresolution algorithms under local translation. IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(1), 83–97 (2004)
6. Nah, S., Baik, S., Hong, S., Moon, G., Son, S., Timofte, R., Mu Lee, K.: Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)
7. Ranjan, A., Black, M.J.: Optical flow estimation using a spatial pyramid network. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 4161–4170 (2017)
8. Rim, J., Lee, H., Won, J., Cho, S.: Real-world blur dataset for learning and benchmarking deblurring algorithms. In: European Conference on Computer Vision. pp. 184–201 (2020)
9. Robinson, M.D., Toth, C.A., Lo, J.Y., Farsiu, S.: Efficient fourier-wavelet super-resolution. IEEE Transactions on Image Processing **19**(10), 2669–2681 (2010)
10. Sfeir, R., Chebaro, B., Julien, C.: Fast 3d volume super resolution using an analytical solution for l2-l2 problems (2020)
11. Šroubek, F., Kamenickỳ, J., Milanfar, P.: Superfast superresolution. In: IEEE International Conference on Image Processing. pp. 1153–1156 (2011)
12. Yang, J., Wright, J., Huang, T.S., Ma, Y.: Image super-resolution via sparse representation. IEEE Transactions on Image Processing **19**(11), 2861–2873 (2010)
13. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: International Conference on Curves and Surfaces. pp. 711–730 (2010)
14. Zhang, K., Gool, L.V., Timofte, R.: Deep unfolding network for image super-resolution. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 3217–3226 (2020)
15. Zhao, N., Wei, Q., Basarab, A., Dobigeon, N., Kouamé, D., Tourneret, J.Y.: Fast single image super-resolution using a new analytical solution for l2-l2 problems. IEEE Transactions on Image Processing **25**(8), 3683–3697 (2016)
16. Zhu, X., Hu, H., Lin, S., Dai, J.: Deformable convnets v2: More deformable, better results. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 9308–9316 (2019)