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

Jiezhang Cao\textsuperscript{1}, Jingyun Liang\textsuperscript{1}, Kai Zhang\textsuperscript{1*}, Wenguan Wang\textsuperscript{1}, Qin Wang\textsuperscript{1}, Yulun Zhang\textsuperscript{1}, Hao Tang\textsuperscript{1}, and Luc Van Gool\textsuperscript{1,2}

\textsuperscript{1}Computer Vision Lab, ETH Zürich, Switzerland  \textsuperscript{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 $H$ and a decimation matrix $S$, then we can rewrite the video degradation model in the paper as follows:

$$Y = (X \otimes K) \downarrow_s + N$$
$$\Rightarrow y = SHx + n,$$

where $y = \text{vec}(Y)$, $x = \text{vec}(X)$ and $n = \text{vec}(N)$. Here, $\text{vec}(\cdot)$ 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 $H$ is the matrix representation of the cyclic convolution, i.e., $H$ 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

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

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

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

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

In this assumption, the decimation matrix satisfies $SS^* = I$.

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

$$
\min_X E(X) := \frac{1}{2\sigma^2} \| Y - (X \otimes K) \downarrow_s \|^2 + \lambda \Phi(X),
$$

(S4)

$$
\Rightarrow \min_x E(x) := \frac{1}{2\sigma^2} \| y - SHx \|^2 + \lambda \Phi(x).
$$

(S5)

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

$$
E(x, z) := \frac{1}{2\sigma^2} \| y - SHx \|^2 + \lambda \Phi(x) + \frac{\mu}{2} \| z - x \|^2.
$$

(S6)

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

$$
\begin{align}
  z_k &= \arg \min_z \| y - SHz \|^2 + \mu \sigma^2 \| z - x_{k-1} \|^2, \\
  x_k &= \arg \min_x \frac{\mu}{2} \| z_k - x \|^2 + \lambda \Phi(x).
\end{align}
$$

(S7)

(S8)

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

$$
z_k = \left( H^* S^* S H + \mu \sigma^2 I \right)^{-1} \left( H^* S^* y + \mu \sigma^2 x_{k-1} \right).
$$

(S9)

However, the solution Eq. (S9) requires the inversion of a high dimensional matrix, whose computational complexity is $O(T^3h^3W^3h^3)$. Based on [15], we extend the case of 2D to 3D, and the decimation matrix $S$ has the following property in the Fourier Domain,

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

$$
FS^*SF^* = \frac{1}{s_w s_h s_t} \left( J_{s_t} \otimes I_t \right) \otimes \left( J_{s_w} \otimes I_w \right) \otimes \left( J_{s_h} \otimes I_h \right),
$$

(S10)

where $I_t \in \mathbb{R}^{t \times t}$ is the an identity matrix and $\otimes$ is the Kronecker product.

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

$$
A^* FS^* F^* A = \frac{1}{s} [A_1, ..., A_s] \top [A_1, ..., A_s] := \frac{1}{s} A^* A.
$$

(S11)
Based on Lemmas 1, 2 and 3, we have

\[ A^* F S^* S^* F^* A \]

\[ = \frac{1}{s} A^*((J_{s_L} \otimes I_t) \otimes (J_{s_w} \otimes I_w) \otimes (J_{s_h} \otimes I_h)A) \]

\[ = \frac{1}{s} A'((1_{s_L} \mathbf{1}^T_{s_L}) \otimes (I_t I_t)) \otimes ((1_{s_w} \mathbf{1}^T_{s_w}) \otimes (I_w I_w)) \otimes ((1_{s_h} \mathbf{1}^T_{s_h}) \otimes (I_h I_h))A \]

\[ = \frac{1}{s} A'((1_{s_L} \otimes I_t)(1_{s_L} \otimes I_t)) \otimes ((1_{s_w} \otimes I_w)(1_{s_w} \otimes I_w)) \otimes ((1_{s_h} \otimes I_h)(1_{s_h} \otimes I_h))A \]

\[ = \frac{1}{s} A'([I_t, ..., I_t] \otimes [I_t, ..., I_t]) \otimes ([I_w, ..., I_w] \otimes [I_w, ..., I_w]) \otimes ([I_h, ..., I_h] \otimes [I_h, ..., I_h])A \]

\[ = \frac{1}{s} A'[I_{twh}, ..., I_{twh}]^T [I_{twh}, ..., I_{twh}]A \]

\[ = \frac{1}{s} A^* A. \]

**Lemma 3.** (Woodbury formula [3]) Given matrices \( A_1, A_2, A_3 \) and \( A_4 \), the following equality holds conditional on the existence of \( A_1^{-1} \) and \( A_3^{-1} \), we have

\[ (A_1 + A_2 A_3 A_4)^{-1} = A_1^{-1} - A_1^{-1} A_2 (A_3^{-1} + A_4 A_1^{-1} A_2)^{-1} A_4 A_1^{-1}. \]  

**(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

\[ z_k = \frac{1}{\mu \sigma^2} r_{k-1} - \frac{1}{\mu \sigma^2} F^* A^* (\mu \sigma^2 s I + A A^*)^{-1} A F r_{k-1}, \]  

where \( s = s_w s_h s_t \), \( r_{k-1} = (H^* S^* y + \mu \sigma^2 x_{k-1}) \).

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

\[ z_k = (F^* A^* F S^* S^* F^* A F + \mu \sigma^2 I)^{-1} F (H^* S^* y + \mu \sigma^2 x_{k-1}) \]  

\[ = F^* \left( \frac{1}{s} A^* A + \mu \sigma^2 I \right)^{-1} F (H^* S^* y + \mu \sigma^2 x_{k-1}) \]  

\[ = F^* \left( \frac{1}{s} A^* A + \mu \sigma^2 I \right)^{-1} F r_{k-1} \]  

\[ = F^* \left( \frac{1}{\mu \sigma^2 I} - \frac{1}{\mu \sigma^2} A^* \left( s I + \frac{1}{\mu \sigma^2} A A^* \right)^{-1} A \frac{1}{\mu \sigma^2} \right) F r_{k-1} \]  

\[ = \frac{1}{\mu \sigma^2} r_{k-1} - \frac{1}{\mu \sigma^2} F^* A^* (\mu \sigma^2 s I + A A^*)^{-1} A F r_{k-1}, \]  

\[ (S13) \]
Eq. (S14) can be directly obtained by taking the derivative of Problem (S7) and make it equal to 0, i.e.,

\[
(F^* \Lambda^* F S^* S F^* \Lambda F + \mu \sigma^2 I) z_k - (H^* S^* y + \mu \sigma^2 x_{k-1}) = 0 \quad (S19)
\]

\[
\Rightarrow z_k = F^* (\Lambda^* F S^* S F^* \Lambda + \mu \sigma^2 I)^{-1} F (H^* S^* y + \mu \sigma^2 x_{k-1}) . \quad (S20)
\]

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

\[
\Lambda^* F S^* S F^* \Lambda = \frac{1}{s} \Lambda^* \Lambda . \quad (S21)
\]

Eq. (S16) can be rewritten by using

\[
r_{k-1} = (H^* S^* y + \mu \sigma^2 x_{k-1}) . \quad (S22)
\]

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

\[
Z_k = F^{-1} \left( \frac{1}{\alpha_k} \left( F(R_{k-1}) - \overline{F}(K) \left( \frac{(F(K)F(R_{k-1})) \uparrow_s}{s \alpha_k I + (F(K)F(K)) \downarrow_s} \right) \right) \right) , \quad (S23)
\]

where \( R_{k-1} = \overline{F}(K)F(Y \uparrow_s) - \alpha_k X_{k-1} \) 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, \( \uparrow_r \) is an upsampler by repeating the tensor the desired dimension, and \( \downarrow_s \) is a distinct block downsampler, i.e., averaging the \( s_t \times s_s \times s_w \) 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 \( T_h W_h H_h \) (here, we ignore the constant number of channels), the computation complexity of Eq. (S23) is \( O(T_h W_h H_h \log(T_h W_h H_h)) \) [10], which is much smaller than the complexity of Eq. (S9) (i.e., \( O(T_h^3 W_h^3 H_h^3) \)) and can be computed efficiently on the modern GPU devices.
B More Details of RVE

We implement $G$ 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 $g_k^i$, we use the optical-flow-guided deformable alignment as $G$ to compute the features at every branch, i.e.,

$$
\bar{z}_k^i = G(g_k^i, \bar{z}_k^{i-1}, \bar{z}_k^{i-2}, f_k^{i-1}, f_k^{i-2}),
$$

$$
= DCN([\bar{z}_k^{i-1}; \bar{z}_k^{i-2}], [o_k^{i-1}; o_k^{i-2}], [m_k^{i-1}; m_k^{i-2}]),
$$

where $[\cdot; \cdot]$ is the concatenation along channel dimension, the offsets $o_k^{i-1}$ and $o_k^{i-2}$ and masks $m_k^{i-1}$ and $m_k^{i-2}$ are formulated as

$$
o_k^{i-p} = f_k^{i-p} + \text{Conv}(g_k^i, \bar{z}_k^{i-1}, \bar{z}_k^{i-2}),
$$

$$
m_k^{i-p} = \text{Sigmoid}(\text{Conv}(g_k^i, \bar{z}_k^{i-1}, \bar{z}_k^{i-2})),
$$

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

$$
\bar{z}_k^{i-1} = \text{warp}(\bar{z}_k^{i-1}, f_k^{i-1}),
$$

$$
\bar{z}_k^{i-2} = \text{warp}(\bar{z}_k^{i-2}, f_k^{i-2}),
$$

where $\text{warp}(\cdot)$ 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 $2.5 \times 10^{-5}$. In the FAT layer, we use torch.rfft and torch.irfft to calculate FFT and inverse FFT operators, respectively. At the beginning, $\mu_k$ is initialized as a small value to accelerate the convergence and then increased gradually during optimization. In the RVE layer, $\beta_k$ decreases as the iteration increases, whose setting is the same as [14]. At the iteration $k=0$, we initialize $X_0$ by interpolating $Y$ with scale factor $s$ 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.
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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Model size (M)</th>
<th>Inference time (s)</th>
<th>PSNR/SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIN+EDVR</td>
<td>25.3</td>
<td>0.658</td>
<td>23.52/0.662</td>
</tr>
<tr>
<td>BIN+BasicVSR</td>
<td>11.0</td>
<td>0.343</td>
<td>24.06/0.678</td>
</tr>
<tr>
<td>BIN+IconVSR</td>
<td>13.4</td>
<td>0.350</td>
<td>24.24/0.687</td>
</tr>
<tr>
<td>BIN+BasicVSR++</td>
<td>12.0</td>
<td>0.357</td>
<td>24.44/0.711</td>
</tr>
<tr>
<td>CDVD-TSP+Zooming Slow-Mo</td>
<td>27.3</td>
<td>3.460</td>
<td>25.99/0.759</td>
</tr>
<tr>
<td>EDVR+Zooming Slow-Mo</td>
<td>31.7</td>
<td>0.438</td>
<td>26.12/0.764</td>
</tr>
<tr>
<td>DAIN+CDVD-TSP+SwinIR</td>
<td>52.1</td>
<td>-</td>
<td>25.01/0.720</td>
</tr>
<tr>
<td>DAIN+CDVD-TSP+EDVR</td>
<td>60.8</td>
<td>4.104</td>
<td>25.38/0.737</td>
</tr>
<tr>
<td>DAIN+CDVD-TSP+BasicVSR</td>
<td>46.5</td>
<td>3.788</td>
<td>25.61/0.748</td>
</tr>
<tr>
<td>DAIN+CDVD-TSP+IconVSR</td>
<td>48.9</td>
<td>3.796</td>
<td>25.74/0.753</td>
</tr>
<tr>
<td>DAIN+CDVD-TSP+BasicVSR++</td>
<td>47.5</td>
<td>3.803</td>
<td>25.90/0.758</td>
</tr>
<tr>
<td>DAIN+EDVR†+SwinIR</td>
<td>56.5</td>
<td>-</td>
<td>25.43/0.730</td>
</tr>
<tr>
<td>DAIN+EDVR†+EDVR</td>
<td>65.2</td>
<td>1.082</td>
<td>25.78/0.747</td>
</tr>
<tr>
<td>DAIN+EDVR†+BasicVSR</td>
<td>50.9</td>
<td>0.767</td>
<td>25.81/0.748</td>
</tr>
<tr>
<td>DAIN+EDVR†+IconVSR</td>
<td>53.3</td>
<td>0.774</td>
<td>25.89/0.752</td>
</tr>
<tr>
<td>DAIN+EDVR†+BasicVSR++</td>
<td>51.9</td>
<td>0.781</td>
<td>25.98/0.755</td>
</tr>
<tr>
<td>DAIN+EDVR†</td>
<td>44.6</td>
<td>0.704</td>
<td>25.77/0.747</td>
</tr>
<tr>
<td>DAIN+BasicVSR*</td>
<td>30.3</td>
<td>0.389</td>
<td>26.08/0.763</td>
</tr>
<tr>
<td>DAIN+IconVSR*</td>
<td>32.7</td>
<td>0.396</td>
<td>26.24/0.768</td>
</tr>
<tr>
<td>DAIN+BasicVSR++*</td>
<td>31.3</td>
<td>0.403</td>
<td>26.53/0.778</td>
</tr>
<tr>
<td>SuperSloMo+CDVD-TSP+SwinIR</td>
<td>47.9</td>
<td>-</td>
<td>25.26/0.726</td>
</tr>
<tr>
<td>SuperSloMo+CDVD-TSP+EDVR</td>
<td>56.6</td>
<td>4.058</td>
<td>25.62/0.739</td>
</tr>
<tr>
<td>SuperSloMo+CDVD-TSP+BasicVSR</td>
<td>42.3</td>
<td>3.743</td>
<td>25.88/0.752</td>
</tr>
<tr>
<td>SuperSloMo+CDVD-TSP+IconVSR</td>
<td>44.7</td>
<td>3.750</td>
<td>26.04/0.756</td>
</tr>
<tr>
<td>SuperSloMo+CDVD-TSP+BasicVSR++</td>
<td>43.3</td>
<td>3.757</td>
<td>26.24/0.762</td>
</tr>
<tr>
<td>SuperSloMo+EDVR†+SwinIR</td>
<td>52.3</td>
<td>-</td>
<td>25.60/0.732</td>
</tr>
<tr>
<td>SuperSloMo+EDVR†+EDVR</td>
<td>61.0</td>
<td>1.036</td>
<td>26.05/0.751</td>
</tr>
<tr>
<td>SuperSloMo+EDVR†+BasicVSR</td>
<td>46.7</td>
<td>0.721</td>
<td>26.06/0.753</td>
</tr>
<tr>
<td>SuperSloMo+EDVR†+IconVSR</td>
<td>49.1</td>
<td>0.728</td>
<td>26.16/0.756</td>
</tr>
<tr>
<td>SuperSloMo+EDVR†+BasicVSR++</td>
<td>47.7</td>
<td>0.735</td>
<td>26.20/0.759</td>
</tr>
<tr>
<td>SuperSloMo+EDVR*</td>
<td>40.4</td>
<td>0.658</td>
<td>26.05/0.752</td>
</tr>
<tr>
<td>SuperSloMo+BasicVSR*</td>
<td>26.1</td>
<td>0.343</td>
<td>26.41/0.769</td>
</tr>
<tr>
<td>SuperSloMo+IconVSR*</td>
<td>82.5</td>
<td>0.350</td>
<td>26.60/0.775</td>
</tr>
<tr>
<td>SuperSloMo+BasicVSR++*</td>
<td>27.1</td>
<td>0.357</td>
<td>26.93/0.785</td>
</tr>
<tr>
<td>Ours</td>
<td>8.0</td>
<td>0.29</td>
<td>29.12/0.859</td>
</tr>
</tbody>
</table>
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 \cite{REDS}. Note that the commonly used blur dataset (i.e., REDS \cite{REDS}) 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 \cite{8}. Without knowing the underlying kernels, our model is able to remove the bluriness, and outperforms the baseline (DAIN+EDVR+BasicVSR++).

Table B: Comparison of other kernels on REDS4.

<table>
<thead>
<tr>
<th>Methods</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/ dynamic blur</td>
<td>28.78</td>
</tr>
<tr>
<td>w/ dynamic and Gaussian/disk blur</td>
<td>28.92</td>
</tr>
</tbody>
</table>

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.

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.

<table>
<thead>
<tr>
<th>Order</th>
<th>PSNR/SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFI→VD→VSR</td>
<td>26.26/0.759</td>
</tr>
<tr>
<td>VD→VFI→VSR</td>
<td>26.24/0.759</td>
</tr>
<tr>
<td>VSR→VFI→VD</td>
<td>26.23/0.756</td>
</tr>
<tr>
<td>Ours</td>
<td>29.12/0.859</td>
</tr>
</tbody>
</table>

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 \(n_{\text{train}}\) and \(n_{\text{test}}\)) 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 \(n_{\text{train}}\) and \(n_{\text{test}}\).

<table>
<thead>
<tr>
<th>#frames</th>
<th>(n_{\text{test}}=5)</th>
<th>(n_{\text{test}}=10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n_{\text{train}}=4)</td>
<td>28.40/0.848</td>
<td>29.02/0.862</td>
</tr>
<tr>
<td>(n_{\text{train}}=5)</td>
<td>28.81/0.857</td>
<td>29.12/0.859</td>
</tr>
<tr>
<td>(n_{\text{train}}=6)</td>
<td>28.89/0.860</td>
<td>29.17/0.862</td>
</tr>
</tbody>
</table>
References