A EVENTS GENERATION

The event stream E consists of a set of event $e = (x, y, t, p)$, where each event is triggered and recorded when the brightness change at pixel (x, y) exceeds a certain threshold C. The time interval between events is denoted as Δt_e, which is a short period, and the brightness at position $\boldsymbol{x} = (x, y)$ is represented as $F(\boldsymbol{x}, t)$. The brightness change can be calculated as $\Delta b = log(F(\boldsymbol{x}, t_e)) - log(F(\boldsymbol{x}, t_e - \Delta t_e))$. The output signal p is determined by Eq. 8.

$$p = \begin{cases} 1, & \Delta b > C \\ 0, & others \\ -1, & \Delta b < -C \end{cases} \tag{8}$$

B THE ENCODING NETWORK STRUCTURE

The encoder serves as the core component of our architecture, drawing inspiration from eSL-Net(Wang et al., 2020). However, we have made modifications to its structure by excluding the decoding segment responsible for upsampling. Consequently, we retain solely the feature extraction module, as illustrated in Fig. 11, and the code of the encoder is shown in Code (Listing. 1).

The encoder receives inputs, rolling shutter blur image I_{rsb}, and events E. The image I_{rsb} has the shape of $H \times W \times 1$ or $H \times W \times 3$, corresponding to grayscale and RGB image, respectively. Moreover, the event E is transformed into count images (Zheng et al., 2023), with the shape of $H \times W \times M$, where M denotes the number of temporal divisions within the event stream. The encoder produces a high-dimensional tensor as output θ, with the shape of $H \times W \times C$.

The encoder can be decomposed into two constituent components: data preprocessing and spatio-temporal information modeling. During the preprocessing stage, the image undergoes a convolution operation to augment the channel dimensionality, while the event data is transformed into a high-dimensional Tensor using two convolutions followed by a Sigmoid activation. Subsequently, the processed tensors from the image and event data are subjected to the sparse learning module. Within the sparse learning module, both image features and event features undergo iterative cycles to derive spatio-temporal representations θ. In contrast to the original approach eSL-Net (Wang et al., 2020), we aim to incorporate deformable convolutions (Wang et al., 2022a) into this loop, thereby enhancing the motion estimation and correction capabilities throughout the iterations.

C MORE EXPLANATION AND DISCUSSION

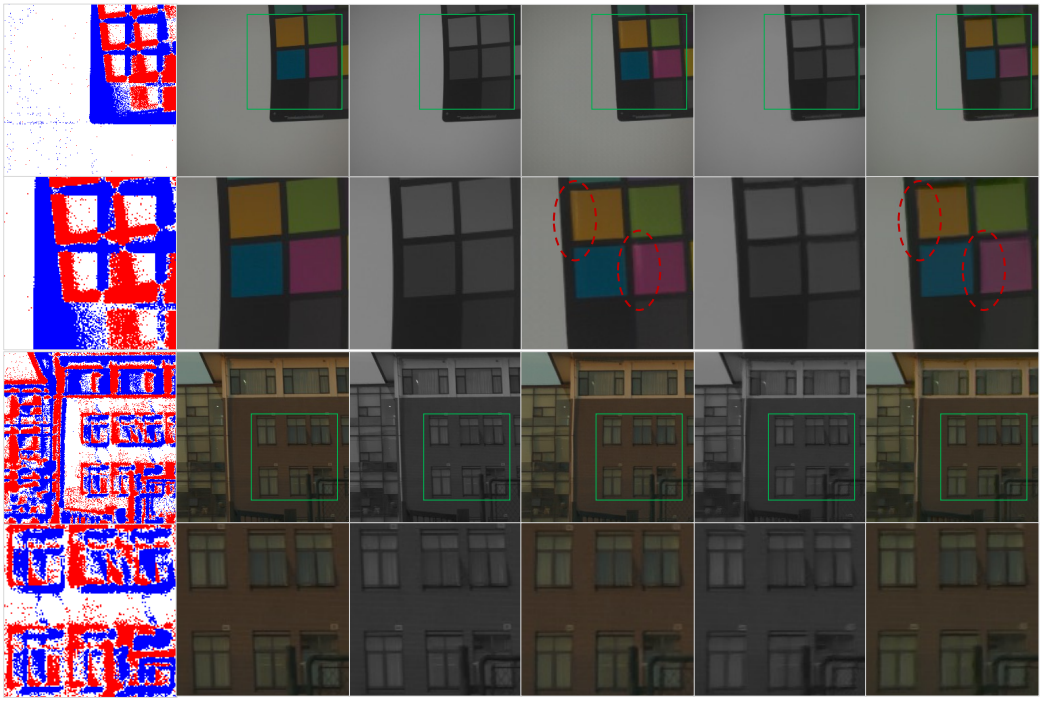
Due to the constraints of the main paper's length, this supplementary section provides additional experimental details and discussions. Specifically, we elaborate on the following 11 aspects to offer readers a more comprehensive understanding of our approach:

C.1 FURTHER DETAIL ON THE DATASET:

1) Gev-RS dataset (Zhou et al., 2022) contains original videos shot by GS high-speed cameras with 1280×720 resolution at 5700 fps. However, EvUnroll (Zhou et al., 2022) primarily focuses on RS correction, and provided by EvUnroll Gev-RS dataset does not include RS frames with severe motion blur. Therefore, we reconstruct RS frames with severe motion blur and events from original videos. We initially downsample the original videos to DAVIS346 event camera's resolution (260×346) (Scheerlinck et al., 2019). Then, we employ the event simulator vid2e (Gehrig et al., 2020) to synthesize events from the resized frames. We simulate RS blur frames by first generating RS sharp frames as the same RS simulation process of Fastec-RS (Liu et al., 2020) and then averaging 260 RS sharp frames after gamma correction. We use the same dataset split as EvUnroll (Zhou et al., 2022), with 20 videos used for training and 9 videos used for testing. The total amounts of RS blur frames in Gev-RS (Zhou et al., 2022) dataset are 784 in the training set and 441 testing set.

2) Fastec-RS dataset (Liu et al., 2020) provides the original frame sequences recorded by the high-speed GS cameras with the resolution of 640×480 at 2400 fps. We use the same settings to resize frame sequences, create events, and RS blurry frames. Furthermore, we use the same dataset split strategy as Fastec-RS (Liu et al., 2020): 56 sequences for training and 20 sequences for testing. Specifically, this dataset includes 1620 RS blur frames for training and 636 RS blur frames for testing.

3) Real-world dataset (Zhou et al., 2022) currently the sole real dataset accessible, comprises four videos. Among these, two capture outdoor scenes, while the other two focus on indoor scenes. Each video pairs rolling shutter frames with events; the events are derived from DVS346. However, given the absence of ground truth in this dataset, it can only provide quantitative visualization results.



(a) Events (b) Color (c) Gray (d) EvUnRoll (e) Ours Gray (f) Our Color

Figure 6: Visualization results in a real-world dataset (Zhou et al., 2022). (a) is the events visualization results. (b) (c) are the input RGB and gray images that have clear rolling shutter distortions. (d) is the output of EvUnRoll. (e) (f) are the outputs of our method.

C.2 EXTENDED DISCUSSION ON INFERENCE SPEED:

Fig. 5 illustrates the inference time of our method with a wide range of interpolation multiples spanning from $1\times$ to $31\times$, including the total inference time and the average inference time per

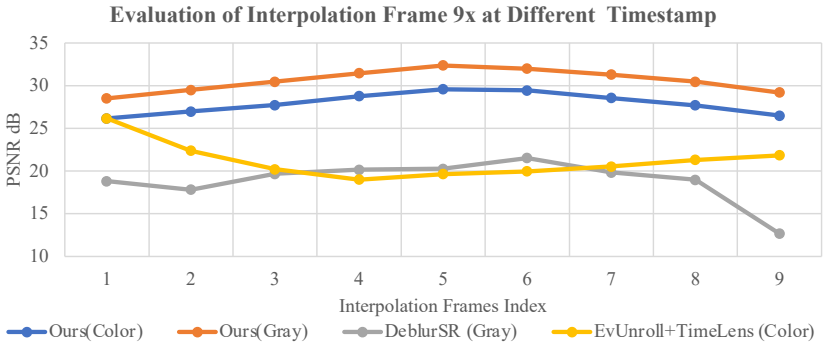


Figure 7: The x-axis corresponds to the subscript of the frame interpolation result achieved through 9-fold interpolation, while the y-axis represents the PSNR value associated with each frame. A higher PSNR value indicates a higher reconstruction quality.

frame. Importantly, the total inference time increases gradually as the frame interpolation multiple increases. For instance, when going from $1\times$ to $31\times$ frame interpolation, the total inference time only increases from 30.8 ms to 86.9 ms. This signifies a mere 2.8-fold increase in time despite a 31-fold increase in the interpolation multiple. Additionally, it is notable that the average inference time per frame decreases with higher frame interpolation multiples. At $31\times$ frame interpolation, the average time per frame is a mere 2.8 ms.

Our method exhibits distinct advantages over the EvUnRoll (Zhou et al., 2022) and TimeLens (Tulyakov et al., 2021) cascade approaches, particularly in terms of computational efficiency. Specifically, when the focus is solely on RS frame correction and deblurring, the inference time for EvUnRoll is measured at 42.3 ms, while our approach necessitates only 72% of that time. This computational advantage becomes even more pronounced during high-magnification frame interpolation. For instance, in scenarios requiring N-times interpolation, the cascading strategy calls for two invocations of EvUnRoll and $(N-2)$ of TimeLens, with the latter having a time cost of $(t_{TL} = 186.76\ ms)$. Consequently, our method offers a significant advantage in high-magnification frame interpolation scenarios. It is crucial to note that our inference time calculations are restricted to GPU-based computations, intentionally omitting the time required for data loading and storage. In practical applications, the EvUnRoll and TimeLens cascade introduces additional disk I/O overhead, thereby further exacerbating its time consumption.

C.3 Additional Insights into the Real-World Dataset:

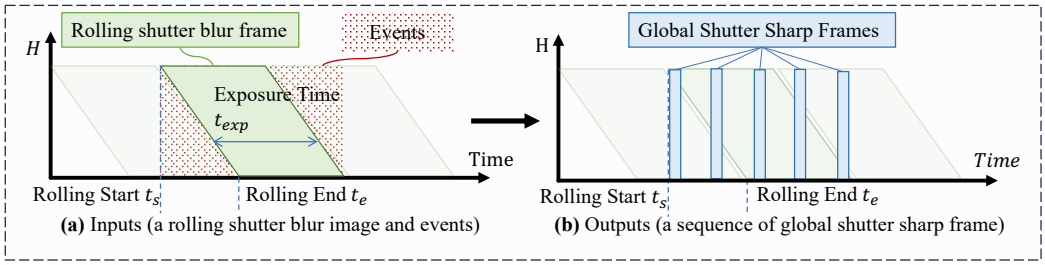
The visualization results for the real-world dataset can be seen in Fig. 6. The input frame, which displays a rolling shutter pattern, is characterized by clear distortions in dynamic scenes. For example, the palette's edges are curved, and the building windows tilt. In contrast, events display global shutter characteristics, as evidenced by the lack of distorted edges in the event visualizations. Both our method and EvUnRoll effectively correct the rolling shutter distortion, whether it's the distortion of the palette's edge or the deformation of building windows. However, due to the absence of ground truth, quantitative analysis remains unattainable. It's worth noting that while EvUnRoll exhibits some artifacts in the palette scenarios, our method remains artifact-free. By concurrently addressing RSC, Deblur, and VFI, our method avoids accumulating errors, leading to a more artifact-free outcome.

C.4 Correlation Between PSNR and Interpolation Frame Index

Fig. 7 illustrates the PSNR values for each frame obtained through various methods using 9-fold frame interpolation. Our proposed method demonstrates superior performance. Specifically, the intermediate image attains the highest quality, while the reconstruction quality diminishes towards both the beginning and end of the exposure, displaying a symmetrical pattern. To further enhance image quality across the entire frame, future investigations could explore the integration of multi-frame algorithms.

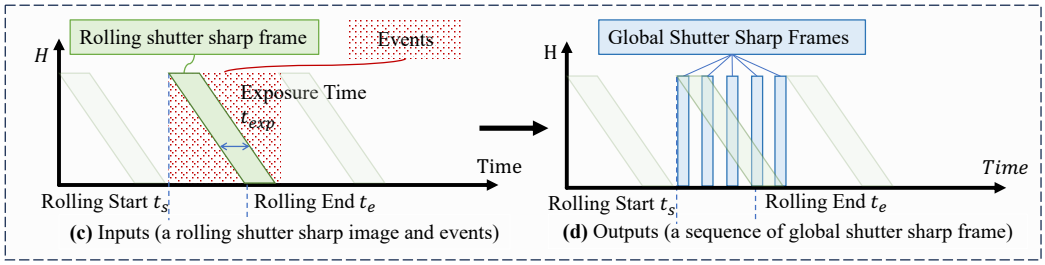
Table 5: More operation studies for exposure time embedding. (Gray blur frame as inputs in 1, 3, 5, 9 times frame interpolation).

	Time Embedding Type	PSNR	SSIM
1×	Add	*33.12*	**0.9881**
	Multiplication	**33.15**	0.9757
	Concat	**33.15**	*0.9876*
3×	Add	*31.11*	**0.9738**
	Multiplication	31.10	0.9635
	Concat	**31.14**	*0.9710*
5×	Add	30.84	**0.9673**
	Multiplication	**30.96**	*0.9684*
	Concat	*30.89*	0.9632
9×	Add	30.54	*0.9579*
	Multiplication	*30.74*	**0.9592**
	Concat	**30.77**	0.9538



(a) Inputs (a rolling shutter blur image and events) **(b)** Outputs (a sequence of global shutter sharp frame)

(I) Long exposure time frame as input. (Rolling shutter correction, Deblur and VFI)



(c) Inputs (a rolling shutter sharp image and events) **(d)** Outputs (a sequence of global shutter sharp frame)

(II) Short exposure time frame as input. (Rolling shutter correction and VFI)

Figure 8: The schematic diagram elucidates the methodologies for correcting and interpolating rolling shutter (RS) frames under varying exposure durations. Subfigure (I) delineates the procedure for long-exposure RS frames, where the presence of blur is a significant factor to be addressed. In contrast, Subfigure (II) outlines the approach for short-exposure RS frames, thereby eliminating the necessity for deblurring.

C.5 More Operations in Exposure Time Embedding

We perform more experiments on Gev-RS dataset (Zhou et al., 2022) to validate the effect of element-wise addition, multiplication, and concatenation in Eq. 6. The quantitative result is shown in Tab. 5, and we find that concatenation and multiplication have higher PSNR than element-wise addition.

C.6 Comparative Studies Across Different Exposure Times

Our model performs reliably with different exposure times. Tab. 1 highlighted results under long exposure scenarios (blurred rolling shutter frames). Tab. 6 shows our model's efficacy in short exposure scenarios (sharp rolling shutter frames). These findings affirm the model's robustness in different exposure times under the rolling shutter pattern.

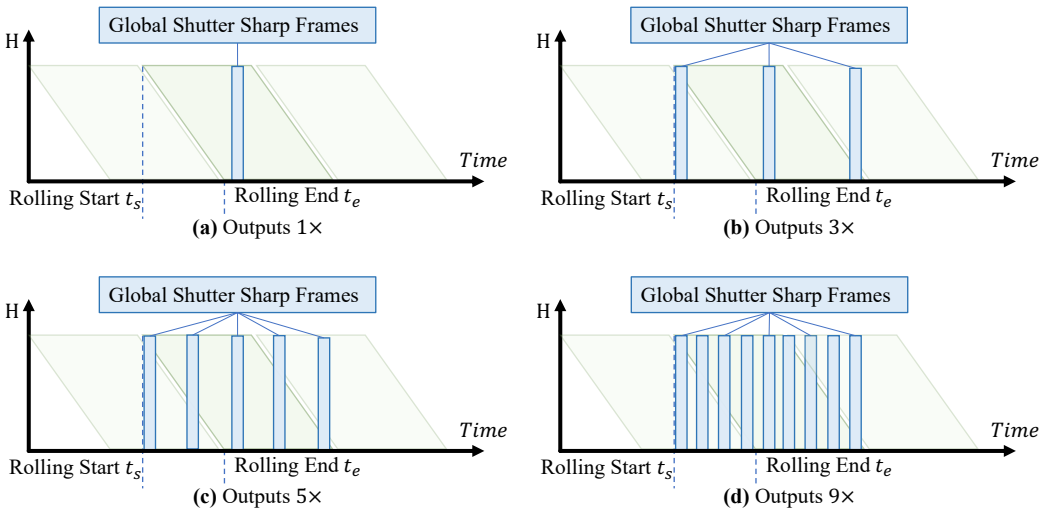


Figure 9: Schematic diagram of frame insertion at different magnifications

Table 6: Experiments on short exposure times.

Method	Exposure Time	Interpolation Frames	PSNR	SSIM
EvUnroll	short	1	28.56	0.9026
Ours	short	1	**29.65**	**0.9457**
Ours	short	3	28.88	0.9364
Ours	short	5	28.89	0.9457
Ours	short	9	28.20	0.9037

C.7 Detailed Comparisons with Other Methods:

Table 7: Comparison of our method with prior works.

Methods	Publication	Frames	Color	Events	Deblur	RS Correction	VFI
JCD	CVPR 2021	3	✓	✗	✓	✓	✗
eSL-Net	ECCV 2020	1	✗	✓	✓	✗	✗
eSL-Net*	ECCV 2020	1	✗	✓	✓	✓	✗
EvUnroll	CVPR 2022	1	✓	✓	✓	✓	-
TimeLens	CVPR 2021	2	✓	✓	✗	✗	✓
E-CIR	CVPR 2022	1	✗	✓	✓	✗	✓
VideoINR	CVPR 2022	2	✓	✗	✗	✗	✓
EvShutter	CVPR 2023	1	✗	✗	✓	✓	✗
DeblurSR	Arxiv 2023	1	✗	✓	✓	✗	✓
NEIR	Arxiv 2023	1	✓	✓	✓	✓	✓
Ours	-	1	✓	✓	✓	✓	✓

In this section, we will explain the motivations for comparing EvUnroll (Zhou et al., 2022) (event-guided RS correction) in the experiment that outputs a single GS sharp frame and comparing EvUnroll + Timelens (Tulyakov et al., 2021) (event-guided video frame interpolation) in the experiment that outputs a sequence of GS sharp frames. Fig. 10 (I) shows the process of generating a sequence of global shutter sharp (GS) frames from a rolling shutter (RS) blur image and paired events by deblur and RS correction. However, the deblur module in EvUnroll recovers the midpoint of the exposure time of each row (Zhou et al., 2022), as shown in Fig. 10 (b); furthermore, EvUnroll can only recover the GS sharp frames between the rolling start time t_s^m and rolling end time t_e^m of the reconstructed RS sharp frame, which can not output the arbitrary GS sharp frames during the whole exposure time of the RS blur frame. Therefore, in the joint task of deblur and RS correction, EvUnroll can not

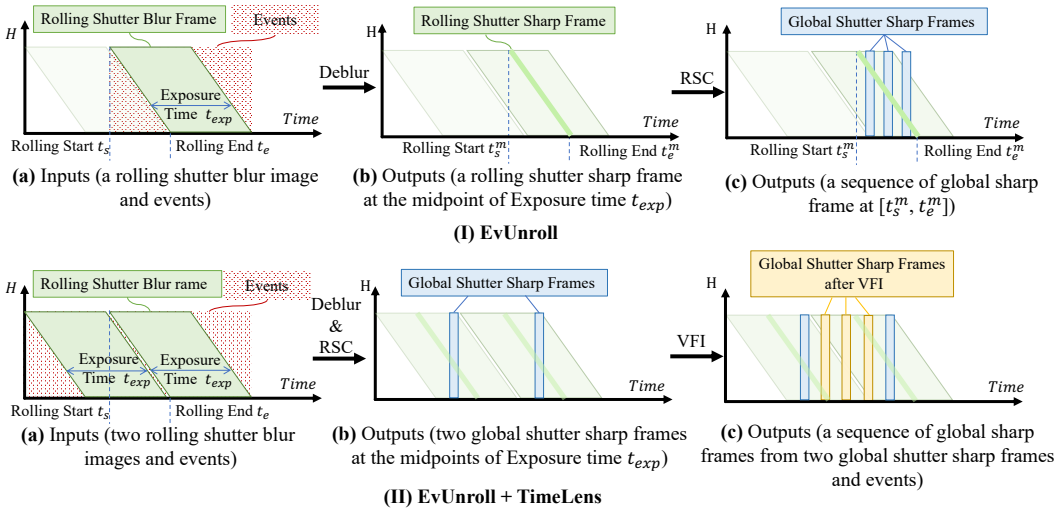


Figure 10: Illustration of Experiment Settings of EvUnroll (Zhou et al., 2022) and the combination of EvUnroll and TimeLens (Tulyakov et al., 2021).

realize arbitrary frame interpolation as shown in Tab. C.7 and we combine EvUnroll and Timelens in the experiment outputting a sequence of GS sharp frames. Specifically, we first generate two GS sharp frames with EvUnroll at the midpoint of the whole exposure time t_{exp} from two RS blur frames and paired events, and then we use TimeLens to generate latent GS sharp frames with the input of two GS sharp frames and events, as shown in Fig. 10 (II).

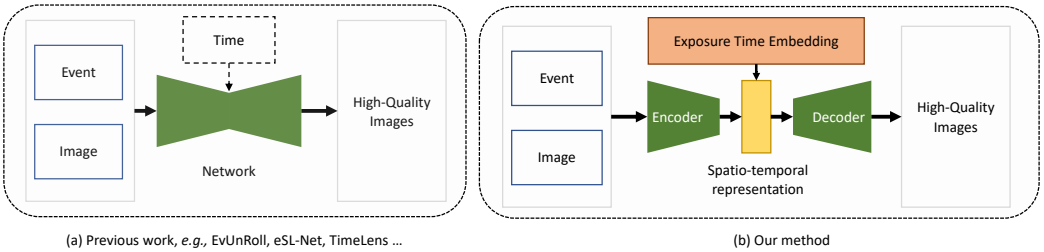
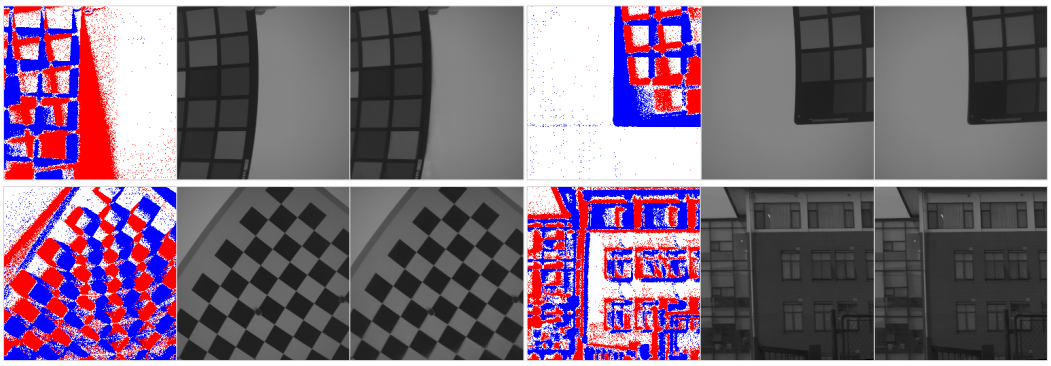


Figure 11: Differences between our method and previous methods. In contrast to the previous method, our approach introduces spatio-temporal representation and exposure time embedding. The spatio-temporal representation involves capturing all the spatio-temporal information during the exposure time. Furthermore, specific exposure time information is embedded, which enables the decoder to generate a frame with high-quality.

Compared with the latest research VideoINR (Chen et al., 2022), our work differs in two aspects. **a)** Different research questions: While VideoINR tackles space-time super-resolution in the global shutter by introducing implicit neural representation (INR), our proposed method first simultaneously realizes RS correction, deblurring, and frame interpolation with INR. **b)** Different methodologies: *i.* VideoINR consists of SpatialINR and TemporalINR, which are sequentially used to transfer the frame feature according to the spatial-temporal coordinate to achieve super-resolution and frame interpolation. However, SpatialINR, and TemporalINR cannot handle motion blur and rolling shutter distortion in the input frames. *ii.* In contrast, our approach develops a unified INR to simultaneously realize RS correction, deblurring, and frame interpolation. Especially, according to the principle of RS and GS images, we design Exposure Time Embedding enabling the generation of RS and GS images given the specific exposure time information, which is a feat unachievable by VideoINR due to its inconsideration towards RS distortion and blur.



(a) Events (b) Input RS Frame (c) Reconstructed RS Frame (d) Events (e) Input RS Frame (f) Reconstructed RS Frame

Figure 12: Rolling shutter frame reconstruction visualization in real-world dataset.

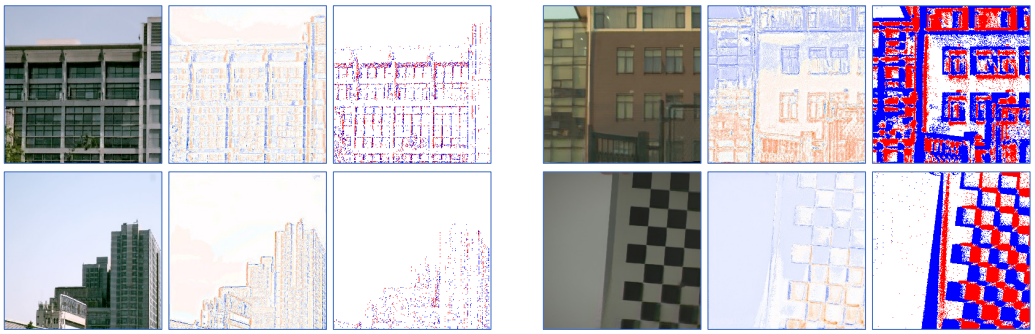


Figure 13: (I) and (I) show visualizations on simulated and real-world datasets, respectively. From left to right: the predicted images, temporal gradients ($\partial F(\boldsymbol{x}, t, \theta)/\partial t$), and events. Orange and blue hues in the image signify positive and negative gradients, respectively. The color intensity is associated with the gradient value, with higher absolute values manifested by stronger colors.

C.8 Visualization of Temporal Dimension Gradients:

Fig. 13 depicts the visualization of the gradients in the temporal dimension, demonstrating the successful training of the function $F(\boldsymbol{x}, t, \theta)$. Both the gradient visualization and events exhibit a similar intensity trend for $F(\boldsymbol{x}, t, \theta)$ at the specified time t. However, the gradient visualization appears smoother with more continuous edges. This observation confirms that our method is capable of learning the high temporal resolution of intensity changes present in events, simultaneously filtering out noise.

C.9 Analysis of PSNR and SSIM Across Different Interpolation Multiples:

In Tab. 1 of the main manuscript and Tab. 6 in supplementary material, we observe an intriguing discrepancy in the PSNR and SSIM metrics for $3\times$ and $5\times$ color frame interpolations, registering values of 28.36 and 0.9062, and 28.41 and 0.9062, respectively. Contrary to conventional wisdom, which posits that an increase in frame rate interpolation should correspondingly degrade PSNR and SSIM metrics when the model architecture remains constant, our findings deviate from this expectation. We attribute this anomaly to the network's varying predictive accuracy across the temporal spectrum. Specifically, edge frames pose a greater challenge for the network compared to those situated centrally. As illustrated in Fig. 7, for a $3\times$ frame insertion, the terminal global

Table 8: Ablation for deformable convolution (DC).

Interpolation multiple	DC	PSNR	SSIM
$1\times$	✗	**33.15**	0.9729
	✓	33.12	**0.9881**
$3\times$	✗	31.11	0.9701
	✓	**31.11**	**0.9738**
$5\times$	✗	30.79	0.9609
	✓	**30.84**	**0.9673**
$9\times$	✗	30.48	**0.9685**
	✓	**30.54**	0.9579
Average Increase		**+ 0.0775**	**+ 0.0088**

shutter sharp frames contribute to $2/3$ of the overall weight. Conversely, for a $5\times$ frame insertion, the terminal frames account for only $2/5$ of the weight.

C.10 Effectiveness of the DCN:

Deformable convolutions offer an effective means of modeling long-range dependencies while preserving computational efficiency. Given these advantages, we have incorporated deformable convolutions into the backbone architecture of encoding. To evaluate their impact, we conducted an ablation study, as presented in Tab. 8. Our results demonstrate an average improvement of 0.0775 dB in PSNR and 0.0088 in SSIM with the inclusion of deformable convolutions, highlighting their beneficial effect.

Due to the limited size of our dataset, we have introduced only one layer of deformable convolution. In contrast to the referenced research (Wang et al., 2022a), which utilized a training set of over 14.2 million samples in the training set, our dataset is comparatively smaller. The GEV training set comprises 784 samples, while the FASTEC training set consists of 1620 samples.

C.11 Exploring Why DeblurSR Appears to Correct Rolling Shutter:

While the primary focus of the DeblurSR (Song et al., 2023) study did not lie in the correction of the rolling shutter effect, we can observe a certain level of correction in the experiments, albeit accompanied by artifacts. We attribute this phenomenon to the fact that events themselves can be viewed as capturing a global shutter perspective. Consequently, the spiking representation learned by DeblurSR using events possesses the potential for rolling shutter correction. The effectiveness of using events to learn implicit representation for rolling shutter correction is evident.

```python
import torch
import torch.nn as nn
from absl.logging import info
from egrsdb.models.unet.dcnv3_nchw import DCNv3NCHW

class _SCN(nn.Module):
    def __init__(self, hidden_channels, high_dim_channels, is_deformable, loop):
        super(_SCN, self).__init__()
        self.hidden_channels = hidden_channels
        self.high_dim_channels = high_dim_channels
        self.is_deformable = is_deformable
        self.loop = loop
        self.W1 = nn.Conv2d(hidden_channels, high_dim_channels, 3, 1, 1, bias=False)
        self.S1 = nn.Conv2d(high_dim_channels, hidden_channels, 3, 1, 1, groups=1, bias=False)
        self.S2 = nn.Conv2d(hidden_channels, high_dim_channels, 3, 1, 1, groups=1, bias=False)
        self.shlu = nn.ReLU(True)
        if is_deformable:
            self.dcn = DCNv3NCHW(channels=high_dim_channels, groups=1, offset_scale=2, act_layer="ReLU", norm_layer="LN", dw_kernel_size=3, center_feature_scale=0.25)

    def forward(self, blur_image, events):
        x1 = blur_image
        event_input = events
        x1 = torch.mul(x1, event_input)
        z = self.W1(x1)
        tmp = z
        for i in range(self.loop):
            ttmp = self.shlu(tmp)
            x = self.S1(ttmp)
            x = torch.mul(x, event_input)
            x = torch.mul(x, event_input)
            x = self.S2(x)
            if self.is_deformable:
                x = torch.relu(x)
                x = self.dcn(x)
            x = ttmp - x
            tmp = torch.add(x, z)
        c = self.shlu(tmp)
        return c


class ESLBackBone(nn.Module):
    def __init__(self, is_color, event_moments, hidden_channels, high_dim_channels, is_deformable, loop):
        super(ESLBackBone, self).__init__()
        in_channel = 3 if is_color else 1
        self.in_channel = in_channel
        self.event_moments = event_moments
        self.hidden_channels = hidden_channels
        self.high_dim_channels = high_dim_channels
        self.is_deformable = is_deformable
        self.loop = loop
        self.image_d = nn.Conv2d(in_channels=in_channel, out_channels=self.hidden_channels, kernel_size=1, stride=1, padding=0, bias=False)
        self.event_c1 = nn.Conv2d(in_channels=event_moments, out_channels=self.hidden_channels, kernel_size=1, stride=1, padding=0, bias=False)
```

```
        self.event_c2 = nn.Conv2d(in_channels=self.hidden_channels,
    out_channels=self.hidden_channels, kernel_size=1, stride=1, padding
    =0, bias=False)
        self.relu = nn.ReLU(inplace=True)
        self.end_conv = nn.Conv2d(in_channels=128, out_channels=
    high_dim_channels, kernel_size=3, stride=1, padding=1, bias=False)
        self.scn_1 = _SCN(hidden_channels, high_dim_channels,
    is_deformable, loop)

    def forward(self, events, blur_frame):
        x1 = self.image_d(blur_frame)
        event_out = self.event_c1(events)
        event_out = torch.sigmoid(event_out)
        event_out = self.event_c2(event_out)
        event_out = torch.sigmoid(event_out)
        out = self.scn_1(x1, event_out)
        out = self.end_conv(out)
        return out
```

Listing 1: The pytorch implementation of encoder.