

# Gyro-based Deep Video Deblurring

— *Supplementary Material* —

Jaesung Rim<sup>1</sup> Woohyeok Kim<sup>1</sup> Haeyun Lee<sup>2</sup> Heemin Yang<sup>1</sup> Ke Wang<sup>3</sup> Sunghyun Cho<sup>1</sup>  
 POSTECH<sup>1</sup> KOREATECH<sup>2</sup> Pika Labs<sup>3</sup>

In this supplementary material, we present the detailed derivation of our decomposed motion model (Sec. S1), the details of blur kernel construction (Sec. S2), a discussion on object motion (Sec. S3), and additional analyses related to object motion (Sec. S4). We then provide details of the network architecture (Sec. S5), the synthesis process of GyroVD-Syn (Sec. S6), a quantitative comparison of blur kernels (Sec. S7), a comparison using additional metrics (Sec. S8), and a visualization of the KGS-Block (Sec. S9). Finally, we present failure cases of our method (Sec. S10) along with additional qualitative results (Sec. S11). We also provide deblurred video results in the supplementary video.

## S1. Derivation of Decomposed Motion Model

In the main paper, we propose a decomposed motion model based on rigid camera motion. This section presents a more detailed derivation of the motion model. Consider a video frame with an exposure interval  $(t_s, t_e)$ , and let  $\{t_i\}_{i=0}^N$  denote  $N+1$  uniformly spaced samples such that  $t_0 = t_s$  and  $t_N = t_e$ . Suppose that the camera’s rotation matrix  $\mathbf{R}_i$  and translation vector  $\mathbf{t}_i$  at time  $t_i$  are given. A 2D pixel position  $\mathbf{p} = [x, y]^\top$  is warped to a new position  $\mathbf{p}'$  according to the standard projection model:

$$\begin{aligned}\mathbf{p}'_i &= \pi\left(\mathbf{C}\left(\mathbf{R}_i\mathbf{C}^{-1}\mathbf{p}_H + \frac{1}{d}\mathbf{t}_i\right)\right) \\ &= \pi\left(\mathbf{C}\mathbf{R}_i\mathbf{C}^{-1}\mathbf{p}_H + \frac{1}{d}\mathbf{C}\mathbf{t}_i\right)\end{aligned}\quad (\text{S1})$$

where  $\mathbf{C}$  is the camera intrinsic matrix,  $\mathbf{p}_H = [\mathbf{p}^\top, 1]^\top$  is the homogeneous coordinate of  $\mathbf{p}$ ,  $d$  is the depth of the pixel. Here,  $\pi((x, y, w)^\top) = (x/w, y/w)^\top$  denotes the projection from a 3D homogeneous coordinate to a 2D Cartesian coordinate.

The term inside  $\pi(\cdot)$  consists of two parts: the rotational and translational camera motion. In practice, the rotational component can be acquired from gyro data, while the translational component remains unknown and is the one we aim to estimate. However, estimating translation using Eq. (S1) is challenging because it requires the scene depth  $d$ . An alternative is to estimate the translational motion component

in the projected pixel domain, but the nonlinear projection  $\pi(\cdot)$  prevents the warped pixel position from being decomposed into rotational and translational components. Consequently, even with known rotational motion, the translational component cannot be estimated separately. To address this, we introduce a decomposed motion model that enables separate estimation of the two components directly in the pixel domain.

We derive our decomposed motion model by approximating Eq. (S1). By applying a first-order Taylor expansion to the rigid camera motion model, the warped pixel position  $\mathbf{p}'_i$  at time  $t_i$  can be approximated as:

$$\begin{aligned}\mathbf{p}'_i &= \pi\left(\mathbf{C}\mathbf{R}_i\mathbf{C}^{-1}\mathbf{p}_H + \frac{1}{d}\mathbf{C}\mathbf{t}_i\right) \\ &\approx \pi\left(\mathbf{C}\mathbf{R}_i\mathbf{C}^{-1}\mathbf{p}_H\right) + \frac{1}{d}\mathbf{J}\mathbf{C}\mathbf{t}_i \\ &= \pi\left(\mathbf{C}\mathbf{R}_i\mathbf{C}^{-1}\mathbf{p}_H\right) + \boldsymbol{\tau}_i\end{aligned}\quad (\text{S2})$$

where  $\boldsymbol{\tau}_i$  is a translational motion vector defined as  $\boldsymbol{\tau}_i = \frac{1}{d}\mathbf{J}\mathbf{C}\mathbf{t}_i$ .  $\mathbf{J}$  is the Jacobian matrix of the projection function  $\pi((x, y, w)^\top) = (x/w, y/w)^\top$  defined as:

$$\mathbf{J} = \begin{bmatrix} \frac{1}{w} & 0 & -\frac{x}{w^2} \\ 0 & \frac{1}{w} & -\frac{y}{w^2} \end{bmatrix}\quad (\text{S3})$$

The warped pixel position in Eq. (S2) is now decomposed into rotational and translational components in the pixel domain. If the pixel displacement at  $t_i$  can be estimated, the translational motion vector  $\boldsymbol{\tau}_i$  can be obtained by subtracting the rotational motion component. However, estimation of the displacement at  $t_i$  is not directly available because  $t_i$  is an intermediate temporal point within the exposure  $(t_s, t_e)$  of a video frame. To enable estimation of  $\boldsymbol{\tau}_i$  with the optical flows, we assume a constant translational velocity and depths between the two frames. With the constant velocity  $\mathbf{v}$ , the translation at  $t_i$  can be expressed as:

$$\mathbf{t}_i = (t_i - t_s)\mathbf{v}\quad (\text{S4})$$

By substituting  $\mathbf{t}_i$  into Eq. (S2), we obtain our decomposed

motion model:

$$\begin{aligned}\mathbf{p}'_i &\approx \pi(\mathbf{C}\mathbf{R}_i\mathbf{C}^{-1}\mathbf{p}_H) + \frac{t_i - t_s}{d} \mathbf{J}\mathbf{C}\mathbf{v} \\ &= \pi(\mathbf{C}\mathbf{R}_i\mathbf{C}^{-1}\mathbf{p}_H) + \frac{t_i - t_s}{t_e - t_s} \frac{t_e - t_s}{d} \mathbf{J}\mathbf{C}\mathbf{v} \quad (\text{S5}) \\ &= \pi(\mathbf{C}\mathbf{R}_i\mathbf{C}^{-1}\mathbf{p}_H) + \frac{t_i - t_s}{t_e - t_s} \boldsymbol{\tau}\end{aligned}$$

where  $\boldsymbol{\tau}$  denotes a translational motion vector, corresponding to the cumulative translation  $(t_e - t_s)\mathbf{v}$  over the exposure interval  $(t_s, t_e)$ . Based on this model, we estimate  $\boldsymbol{\tau}$  using optical flows between consecutive video frames, as described in Sec. 4. of the main paper. Then, the translational motion vector at  $t_i$  can be approximated as  $\boldsymbol{\tau}_i \approx \frac{t_i - t_s}{t_e - t_s} \boldsymbol{\tau}$ .

## S2. Details of Blur Kernel Construction

Given a blurred video frame  $I$  captured over the exposure interval  $(t_s, t_e)$ , we compute warped pixel trajectories over  $N+1$  uniformly spaced time samples  $\{t_i\}_{i=0}^N$  within the interval, and use them as blur kernels. In the main paper, for brevity, we assume that the pixel-wise trajectories start at the beginning of the exposure  $t_s$ . However, most deblurring methods consider the ground-truth frame to correspond to the temporal center of the exposure. To account for this, we compute both the rotational and translational components of the pixel-wise trajectories using the temporal center  $t_c = (t_s + t_e)/2$  as the starting point instead of  $t_s$ . Below, we provide more details.

**Rotational Component** Using the gyro sensor, we obtain angular velocity measurements  $\{\boldsymbol{\omega}_k\}_{k=0}^M$  and corresponding timestamps  $\{\tilde{t}_k\}_{k=0}^M$  over the exposure interval  $(t_s, t_e)$ . We take the temporal center  $t_c = (t_s + t_e)/2$  as the starting time instead of  $t_s$ . The cumulative rotation from  $t_c$  to  $t_i$  is computed as

$$\mathbf{R}_i = \begin{cases} \mathbf{R}(t_i, t_c)^{-1}, & i \in [0, N/2] \\ \mathbf{R}(t_c, t_i), & i \in (N/2, N] \end{cases} \quad (\text{S6})$$

where  $\mathbf{R}(t_1, t_2)$  denotes the rotation integrated from  $t_1$  to  $t_2$ :

$$\mathbf{R}(t_1, t_2) = \prod_{k: t_1 \leq \tilde{t}_k \leq t_2} \exp([\boldsymbol{\omega}_k]_{\times} \Delta \tilde{t}_k), \quad (\text{S7})$$

where the product is taken in chronological order,  $[\cdot]_{\times}$  denotes the skew-symmetric matrix, and  $\Delta \tilde{t}_k$  is the time interval between consecutive measurements. To compute the rotation from  $t_c$  to  $t_i$ , when  $i \in [0, N/2]$ , we obtain the backward rotation by inverting the cumulative rotation from  $t_i$  to  $t_c$ , whereas for  $i \in (N/2, N]$ , we directly compute the forward rotation from  $t_c$  to  $t_i$ .

As in the main paper, we substitute Eq. (S6) into Eq. (S5) while setting  $\boldsymbol{\tau} = \mathbf{0}$ , and compute the warped position  $\mathbf{p}'_i$

at time  $t_i$  from  $t_c$ . This yields the pixel-wise trajectories induced by rotational camera motion:

$$\mathbf{k}^{\text{rot}} = \{\mathbf{d}_i^{\text{rot}}\}_{i=0}^N, \quad \mathbf{d}_i^{\text{rot}} = \mathbf{p}'_i - \mathbf{p} \quad (\text{S8})$$

where  $\mathbf{d}_i^{\text{rot}}$  denotes the pixel displacement at the  $i$ -th sampling point, and  $\mathbf{k}^{\text{rot}}$  represents the corresponding trajectory formed by  $N$  sampled displacements.

**Translational Component** To compute the translational component, we estimate the translational motion vector. For an input frame  $I$ , we compute optical flow maps from  $I$  to its neighboring frames  $I^{\text{next}}$  and  $I^{\text{prev}}$ , which are assumed to represent the displacement between their temporal centers. To isolate the translational component, we first compute the rotation matrices  $\mathbf{R}^{\text{next}}$  and  $\mathbf{R}^{\text{prev}}$  between the temporal centers of  $I$  and its neighboring frames by accumulating rotations, similar to Eq. (S6). We then estimate two translational motion vectors,  $\boldsymbol{\tau}^{\text{prev}}$  and  $\boldsymbol{\tau}^{\text{next}}$ , corresponding to the motion between  $I$  and its previous and next frames, respectively:

$$\boldsymbol{\tau}^{\text{prev}} = \frac{t_e - t_s}{\delta} \{\mathbf{p}'_{\text{prev}} - \pi(\mathbf{C}\mathbf{R}^{\text{prev}}\mathbf{C}^{-1}\mathbf{p}_H)\} \quad (\text{S9})$$

$$\boldsymbol{\tau}^{\text{next}} = \frac{t_e - t_s}{\delta} \{\mathbf{p}'_{\text{next}} - \pi(\mathbf{C}\mathbf{R}^{\text{next}}\mathbf{C}^{-1}\mathbf{p}_H)\} \quad (\text{S10})$$

where  $\delta$  is the temporal interval between the temporal centers (e.g., 1/30 sec. for 30 FPS videos).  $\mathbf{p}'_{\text{prev}}$  and  $\mathbf{p}'_{\text{next}}$  denote the warped positions of  $\mathbf{p}$  in the previous and next frames, respectively, obtained using the estimated optical flow maps. These two vectors are then distributed across the sampling points  $(t_0, \dots, t_c)$  and  $(t_c, \dots, t_N)$ , respectively, yielding

$$\mathbf{k}^{\text{tran}} = \{\mathbf{d}_i^{\text{tran}}\}_{i=0}^N, \quad \mathbf{d}_i^{\text{tran}} = \frac{|t_i - t_c|}{t_e - t_s} \boldsymbol{\tau}_{\text{prev/next}} \quad (\text{S11})$$

where  $\boldsymbol{\tau}_{\text{prev/next}}$  denotes the estimated translational motion vectors, such that  $\boldsymbol{\tau}_{\text{prev}}$  is used for  $i \in [0, N/2]$  and  $\boldsymbol{\tau}_{\text{next}}$  for  $i \in (N/2, N]$ . Finally, the per-pixel blur kernel can be obtained by combining both components:

$$\mathbf{k} = \mathbf{k}^{\text{rot}} + \mathbf{k}^{\text{tran}} \quad (\text{S12})$$

which captures the complete camera motion trajectory during the exposure interval.

## S3. Discussion on Object Motion

In the main paper, we introduce a decomposed motion model consisting of rotational and translational components of camera motion. Based on this model, we construct blur kernels that reflect both types of camera motion. In dynamic scenes, however, objects move independently from the camera, causing object-induced errors in the constructed blur

kernels. Fortunately, since the GyroVD-Syn dataset naturally contains diverse moving objects with such errors, GyroDVD trained on the dataset learns to handle these errors, leading to improved robustness to object motion. Moreover, we find that our decomposed motion model remains valid under certain assumptions, even in the presence of object motion, which further improves robustness in dynamic scenes. We provide further details below.

Suppose that the camera’s rotation matrix  $\mathbf{R}_i$  and translation vector  $\mathbf{t}_i^{cam}$  at time  $t_i$  are given. For object motion, we additionally consider the translation of each pixel  $\mathbf{t}_i^{obj} \in \mathbb{R}^3$  induced by object motion. Then, the total translation of each pixel is defined as:

$$\mathbf{t}_i = \mathbf{t}_i^{cam} + \mathbf{t}_i^{obj} \quad (\text{S13})$$

By substituting  $\mathbf{t}_i$  into Eq. (S2), the warped pixel position  $\mathbf{p}'_i$  at time  $t_i$  can be expressed as:

$$\begin{aligned} \mathbf{p}'_i &\approx \pi(\mathbf{C}\mathbf{R}_i\mathbf{C}^{-1}\mathbf{p}_H) + \frac{1}{d}\mathbf{J}\mathbf{C}(\mathbf{t}_i^{cam} + \mathbf{t}_i^{obj}) \\ &= \pi(\mathbf{C}\mathbf{R}_i\mathbf{C}^{-1}\mathbf{p}_H) + \boldsymbol{\tau}_i \end{aligned} \quad (\text{S14})$$

where  $\boldsymbol{\tau}_i$  is the translational motion vector defined as  $\boldsymbol{\tau}_i = \frac{1}{d}\mathbf{J}\mathbf{C}(\mathbf{t}_i^{cam} + \mathbf{t}_i^{obj})$ . Here,  $\boldsymbol{\tau}_i$  includes both camera and object translational motion, not just camera motion. Assuming that both the camera and the objects have constant translational velocity within the short interval,  $\boldsymbol{\tau}_i$  can still be approximated by the same formulation as Eq. (S5) and estimated from optical flows using the same procedure described in the main paper. When objects are present in the scene, the optical flows encode the combined displacement from both camera and object motion. By subtracting the rotational component of the camera motion, we obtain a translational motion vector that contains both camera and object translation. As a result, the blur kernels constructed from this vector also reflect object motion.

It is important to note that this modified model is still limited in modeling object motion, as it only accounts for object translation, whereas real-world object motion may involve complex non-rigid deformation or rotation. Nevertheless, we find that this approximation is sufficiently expressive to provide useful cues for handling object motion in practice. A more detailed analysis of object motion is provided in Sec. S4.

## S4. Analysis on Object Motion

In this section, we analyze the performance of GyroDVD with respect to the presence of object motion. To this end, we manually classify all video frames in the test set of GyroVD-Syn according to whether they contain moving objects. Even when moving objects are present, the overall motion usually includes both camera and object motion, making it difficult to analyze the effect of object motion

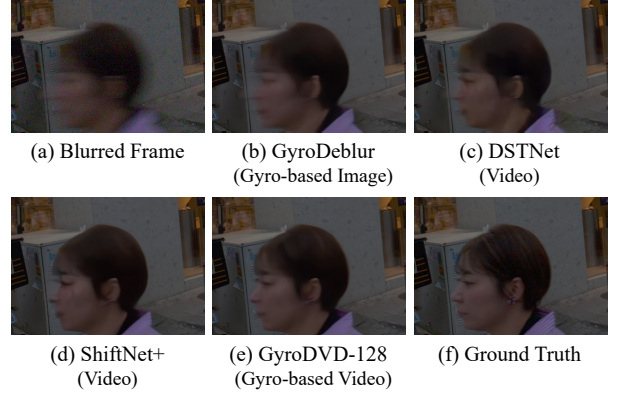


Figure S1. Qualitative results on the object motion only subset of GyroVD-Syn.

alone. Therefore, we additionally classify frames that contain only object motion. Based on these labels, we split the GyroVD-Syn test set into three subsets: camera motion only, camera + object motion, and object motion only, containing 2,967, 4,550, and 183 frames, respectively. We then analyze the effect of object motion on these three subsets.

**Comparison on Object Motion** Tab. S1 shows the deblurring performance of GyroDVD and other methods on the three subsets. The table demonstrates that the performance gap between GyroDVD and the other deblurring methods is the largest in the camera motion only subset, while it becomes smaller in the other subsets. This result indicates that the motion information from gyro data is most beneficial when camera motion is dominant. However, even when moving objects are present in the scene, the resulting motion usually includes both camera and object motion (*i.e.*, the camera + object motion subset). In such cases, the camera motion measured by the gyro data still provides useful cues for deblurring. As a result, GyroDVD achieves superior performance on the other subsets as well. Notably, GyroDVD-128 still outperforms the other methods on the object motion only subset, thanks to the object motion information captured in  $\mathbf{k}^{\text{tran}}$ .

Fig. S1 shows the deblurred results on the object motion only subset. As shown in the figure, GyroDVD successfully restores the face of moving people. Interestingly, GyroDeblur [20] can still handle local object motion, even though the method does not use  $\mathbf{k}^{\text{tran}}$ . This is because gyro-based deblurring networks can learn to handle motion errors caused by moving objects when such errors are present in the training data [20]. Thanks to the diverse moving objects in GyroVD-Syn, GyroDeblur trained on the dataset becomes more robust to object motion. This result highlights the superiority of GyroVD-Syn with realistic camera and object motions in dynamic scenes.

**Ablation Study on Object Motion** To analyze the effect of  $\mathbf{k}^{\text{tran}}$  on object motion, we conduct an additional ablation

Table S1. Quantitative comparison on GyroVD-Syn. We evaluate GyroDVD and other deblurring methods on three subsets: camera motion only, camera + object motion, and object motion only.

	GyroVD-Syn (PSNR $\uparrow$ / SSIM $\uparrow$ )			
	Camera Motion Only	Camera + Object Motion	Object Motion Only	Avg
DeepGyro [13]	28.27 / 0.7555	31.30 / 0.8283	34.00 / 0.8992	30.13 / 0.8003
EggNet [9]	28.47 / 0.7631	31.47 / 0.8338	34.28 / 0.9026	30.32 / 0.8066
GyroDeblur [20]	30.76 / 0.8147	33.31 / 0.8651	35.89 / 0.9183	32.34 / 0.8458
BasicVSR++ [5]	30.81 / 0.8268	34.80 / 0.8968	36.64 / 0.9270	33.22 / 0.8689
EDVR [19]	31.54 / 0.8375	34.44 / 0.8857	36.43 / 0.9234	33.31 / 0.8669
STCT [22]	31.40 / 0.8402	34.64 / 0.8940	36.65 / 0.9289	33.37 / 0.8729
DSTNet [15]	32.06 / 0.8523	35.03 / 0.8995	36.94 / 0.9316	33.86 / 0.8810
ShiftNet [10]	32.76 / 0.8615	35.40 / 0.9026	37.31 / 0.9341	34.37 / 0.8865
VRT [12]	33.05 / 0.8700	35.89 / 0.9109	37.63 / 0.9376	34.77 / 0.8948
RVRT [11]	33.07 / 0.8701	35.97 / 0.9125	37.44 / 0.9350	34.82 / 0.8957
DSTNet+L [16]	33.26 / 0.8755	35.97 / 0.9143	37.60 / 0.9384	34.90 / 0.8990
ShiftNet+ [10]	33.77 / 0.8811	36.31 / 0.9161	38.00 / 0.9403	35.31 / 0.9023
ShiftNet w/ $\mathbf{k}^{\text{rot}}$	32.98 / 0.8662	35.55 / 0.9049	37.34 / 0.9342	34.55 / 0.8898
GyroDVD-48	33.66 / 0.8803	36.07 / 0.9138	37.58 / 0.9370	35.12 / 0.9006
GyroDVD-64	33.95 / 0.8854	36.32 / 0.9172	37.76 / 0.9386	35.39 / 0.9047
GyroDVD-96	34.33 / 0.8917	36.68 / 0.9217	38.10 / 0.9414	35.76 / 0.9099
GyroDVD-128	34.49 / 0.8933	36.86 / 0.9231	38.26 / 0.9423	35.93 / 0.9113

Table S2. Ablation study on blur kernels. We evaluate variants of GyroDVD using alternative blur kernels on three subsets: camera motion only, camera + object motion, and object motion only.

	GyroVD-Syn (PSNR $\uparrow$ / SSIM $\uparrow$ )			
	Camera Motion Only	Camera + Object Motion	Object Motion Only	Avg
Baseline	31.81 / 0.8450	34.59 / 0.8918	36.71 / 0.9289	33.51 / 0.8736
w/ video frame	31.80 / 0.8445	34.59 / 0.8916	36.72 / 0.9288	33.51 / 0.8733
w/ optical flows	32.47 / 0.8589	35.17 / 0.9011	37.02 / 0.9321	34.11 / 0.8846
w/ $\mathbf{k}^{\text{rot}}$ only	32.74 / 0.8629	35.30 / 0.9024	36.95 / 0.9306	34.30 / 0.8870
w/ $\mathbf{k}^{\text{rot}}, \mathbf{k}^{\text{tran}}$ (Ours)	33.12 / 0.8703	35.64 / 0.9073	37.29 / 0.9341	34.65 / 0.8929

study. For this ablation study, variants of GyroDVD-64 are trained for 150K iterations with a batch size of 2. We evaluate the effect of the blur kernels by replacing them with alternative inputs: the video frame itself, bi-directional optical flows, and rotation-only blur kernels using  $\mathbf{k}^{\text{rot}}$ . For these variants, the offsets of the deformable convolutions and the shift patterns are estimated from their respective inputs. We then evaluate these variants on the three subsets: camera motion only, camera + object motion, and object motion only subsets.

Tab. S2 shows the deblurring performance of the variants. The rotation-only model (*i.e.*, w/  $\mathbf{k}^{\text{rot}}$  only) outperforms the variants that use video frames or optical flows when camera motion is present (*i.e.*, the camera motion only and camera + object motion subsets). However, it underperforms on the object motion only subset because blur kernels computed from  $\mathbf{k}^{\text{rot}}$  capture only camera motion. In contrast, our final model achieves superior performance on all subsets, including the object motion only subset, thanks to

the object motion captured in  $\mathbf{k}^{\text{tran}}$ .

Fig. S2 visualizes the blur kernels and deblurred results of the rotation-only model and our final model. As shown in the figure, blur kernels computed solely from  $\mathbf{k}^{\text{rot}}$  are misaligned with the motion of the object. In contrast, the combined kernels constructed from both  $\mathbf{k}^{\text{rot}}$  and  $\mathbf{k}^{\text{tran}}$  more accurately reflect the motion of the object. Consequently, our final model using the combined kernels recovers much sharper details than the rotation-only model.

## S5. Details of Network Architecture

Fig. S3 presents the detailed network architecture of GyroDVD, which consists of three main components: a blur kernel encoding module, an image encoder, and a video decoder. The blur kernel encoding module includes two encoding layers,  $f_{\text{rot}}(\cdot)$  and  $f_{\text{tran}}(\cdot)$ , which extract features from the rotational and translational components of the blur kernels,  $(\mathbf{k}_j^{\text{rot}}, \mathbf{k}_j^{\text{tran}})$ . The extracted features are fused in



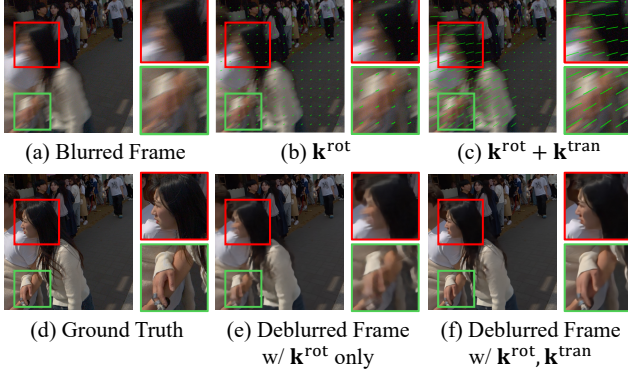


Figure S2. Qualitative results on the object motion only subset of GyroVD-Syn.

the feature domain to produce the kernel features. The initial image features are extracted by  $f_{\text{img}}(\cdot)$ , and the image encoder processes each frame independently. The encoder contains three levels, each consisting of one encoder block. To exploit blur kernels, each encoder block adopts deformable convolution [24], whose offsets and masks are predicted from the blur kernel features. The video decoder aggregates temporal information from the image features with the guidance of the kernel features and produces the final deblurred frames. It also has three levels: each of the first two levels includes 24 Kernel-Guided Shift Blocks (KGS-Blocks), while the last level contains only a channel-attention block following ShiftNet [10]. Each KGS-Block propagates and aggregates features across neighboring frames using learnable shift patterns predicted from the blur kernel features. To reduce computation, these shift patterns are adaptively predicted at every fourth block instead of at every block. Following ShiftNet, we adopt a bi-directional decoding strategy by stacking forward and backward KGS-Blocks alternately. Finally, the decoded features are fed into the reconstruction module  $f_{\text{rec}}(\cdot)$  to generate the deblurred output. The blur kernel encoding layers  $f_{\text{rot}}$  and  $f_{\text{tran}}$ , the image feature extractor  $f_{\text{img}}$ , and the reconstruction module  $f_{\text{rec}}$  each contain six channel-attention blocks [10].

## S6. Details of Synthesizing GyroVD-Syn

As described in the main paper, we adopt a realistic blur synthesis pipeline [17] to improve the robustness of our synthetic dataset on real-world videos. The pipeline consists of three components: frame interpolation, saturation synthesis, and noise synthesis. We provide additional details for each component below.

**Frame Interpolation** Although the source videos used to synthesize GyroVD-Syn are captured at high speed (240 FPS), the temporal gap between frames can still be large when the scene contains large motion. To reduce this gap, we apply frame interpolation before averaging video

frames, following the practice in prior datasets [14, 17, 18]. Using a state-of-the-art interpolation method [21], we increase the frame rate of the source sRGB videos from 240 FPS to 1920 FPS, and subsequently generate blurred frames by averaging the interpolated frames. Additionally, we apply gamma decoding before averaging and gamma encoding afterward to account for the non-linearity of the sRGB space.

**Saturation Synthesis** The averaged frames do not contain saturated pixels because the source sRGB videos are already clipped. To synthesize the saturated pixels, we adopt mask-based saturation synthesis [17] to generate realistic light streaks on video frames.

$$I_{\text{sat}} = \text{clip}(I_{\text{avg}} + \alpha M_{\text{sat}}), \quad (\text{S15})$$

where  $I_{\text{avg}}$  is an averaged video frame and  $M_{\text{sat}}$  is a saturation mask obtained by averaging the saturated pixels in the source frames.  $\text{clip}(\cdot)$  denotes the clipping function, and  $\alpha$  is a scaling factor randomly sampled from a uniform distribution  $\mathcal{U}(0.25, 2.75)$ . We apply the same scaling factor across frames in the same video to ensure temporal consistency of saturated pixels.

**Noise Synthesis** The synthesized blurred frames contain only a small amount of noise because the averaging process reduces noise. In contrast, real-world video frames exhibit a wide range of noise levels depending on the environment and camera settings. Consequently, deblurring networks trained on synthetic datasets without considering realistic noise often fail to generalize to real-world blur [17]. Thus, we synthesize realistic noise to improve the generalization ability of GyroVD-Syn.

Since sensor noise is well modeled in the raw-RGB domain, previous works for noise synthesis [4, 8] convert sRGB images to the raw-RGB space, add Poisson–Gaussian noise in the raw-RGB space, and convert them back to sRGB. Following this, we also convert each frame to the raw-RGB space as:

$$\tilde{I}_{\text{sat}} = \text{sRGB2RAW}(I_{\text{sat}}, M_{\text{CCM}}, M_{\text{WB}}) \quad (\text{S16})$$

where sRGB2RAW is an inverse operation of a simple ISP [17] consisting of gamma decoding, inverse white balance, and color transform from the sRGB space to the raw-RGB space.  $M_{\text{CCM}}$  and  $M_{\text{WB}}$  denote the matrices for color transform and white balance, respectively. For more accurate conversion, we use the matrices extracted from the Android API [1, 2]. These matrices were recorded simultaneously with the source high-speed videos.

To reflect various amounts of noise, we synthesize noise using various ISO levels. We randomly sample the ISO of our camera system (*i.e.*, Google Pixel 9 Pro XL) as:

$$\text{ISO} \sim \mathcal{U}(50, 1600) \quad (\text{S17})$$

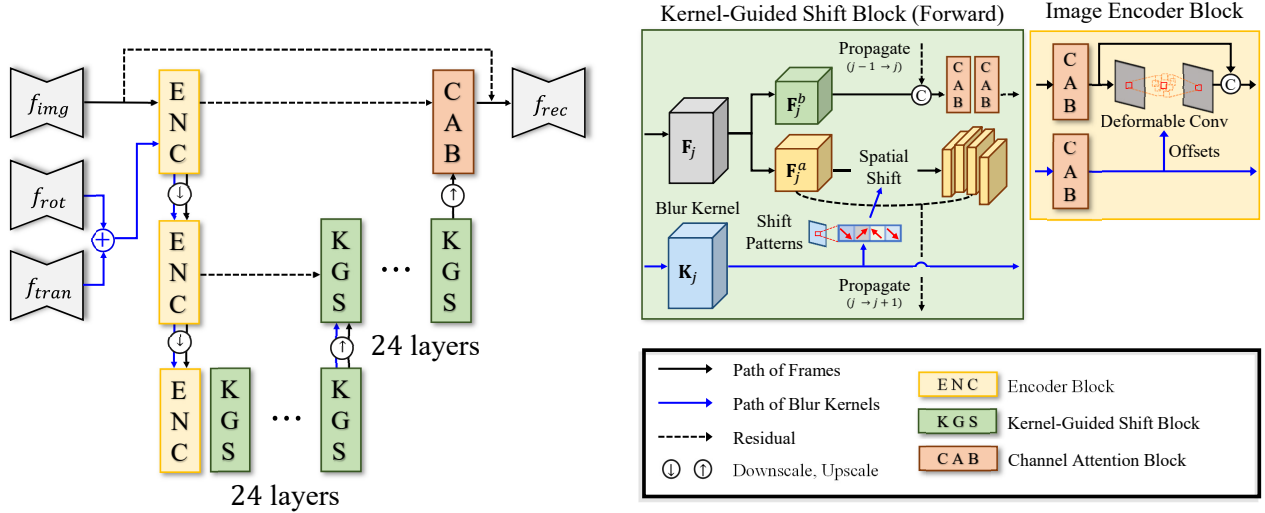


Figure S3. Network Architecture of GyroDVD

where  $\mathcal{U}$  is a uniform distribution. For each ISO, we compute the shot and read noise parameters  $\lambda_{shot}$  and  $\lambda_{read}$  as:

$$\lambda_{shot} = 7.2001e-07 \times \text{ISO} + 1.2589e-05 \quad (\text{S18})$$

$$\log(\lambda_{read}) = 1.4141 \times \log(\lambda_{shot}) - 2.0269 \quad (\text{S19})$$

The multipliers and intercepts are estimated from the calibrated shot and read noise parameters using linear regression. We extracted the calibrated noise parameters from Android noise profile API [3]. A noisy raw frame is then synthesized as:

$$\tilde{I}_{noisy} = (\tilde{I}_{sat} + N_{shot}) + N_{read} \quad (\text{S20})$$

where  $N_{shot}$  and  $N_{read}$  denote Poisson shot noise and Gaussian read noise, respectively:

$$(\tilde{I}_{sat} + N_{shot}) \sim \mathcal{P}\left(\frac{\tilde{I}_{sat}}{\lambda_{shot}}\right) \lambda_{shot} \quad (\text{S21})$$

$$N_{read} \sim \mathcal{N}(0, \lambda_{read}) \quad (\text{S22})$$

To ensure consistent noise levels across a video, the same  $(\lambda_{shot}, \lambda_{read})$  values are applied to all frames within a video. After synthesizing noise in the raw-RGB domain, we convert the noisy frame back to the sRGB space:

$$I_{noisy} = \text{RAW2sRGB}(\tilde{I}_{noisy}, M_{CCM}, M_{WB}) \quad (\text{S23})$$

where RAW2sRGB is a simple ISP consisting of gamma encoding, white balance, and color transform from the raw-RGB space to the sRGB space.

Following Rim *et al.* [17], we apply this pipeline to each video clip on-the-fly during training. We generate the test set of GyroVD-Syn using the same procedure.

Table S3. Quantitative comparison of blur kernels on GyroVD-Syn. We measure  $\ell_1$  distance error between the blur kernels and the ground-truth trajectories computed from the source sharp frames before averaging.

Blur Kernel	$\ell_1$ Error ↓
w/ optical flows	1.0419
w/ $\mathbf{k}^{\text{rot}}$ only	1.1559
w/ $\mathbf{k}^{\text{rot}} + \mathbf{k}^{\text{tran}}$ (Ours)	0.5320
w/ $\mathbf{k}^{\text{rot}} + \mathbf{k}^{\text{tran}}$ from GT	0.4153

## S7. Quantitative Comparison of Blur Kernel

In this section, we evaluate the accuracy of the blur kernels. We compare the proposed blur kernels using  $(\mathbf{k}^{\text{rot}}, \mathbf{k}^{\text{tran}})$  with three variants of blur kernels: blur kernels computed from bi-directional optical flows, rotation-only blur kernels using only  $\mathbf{k}^{\text{rot}}$ , and a variant using  $\mathbf{k}^{\text{tran}}$  computed from ground-truth sharp frames. For the optical flow-based kernels, we use bi-directional optical flows estimated from the current frame to the previous and next frames. These flows capture the displacement between frames and therefore cannot be used directly to describe the motion over the exposure interval. By linearly scaling them according to the exposure time, we convert the flows into vectors that represent the displacement from the temporal center to the start and end of the exposure. We use these two scaled vectors as bi-directional blur kernels. Note that the blur kernels computed from optical flows capture motion only at two temporal points (*i.e.*, from the temporal center to the start and end of the exposure), whereas  $\mathbf{k}^{\text{rot}}$  captures motion at  $N = 8$  sampling points.

To quantitatively evaluate the accuracy of blur kernels, ground-truth blur kernels are required. However, even in the synthetic dataset, the ground-truth blur kernels are not avail-

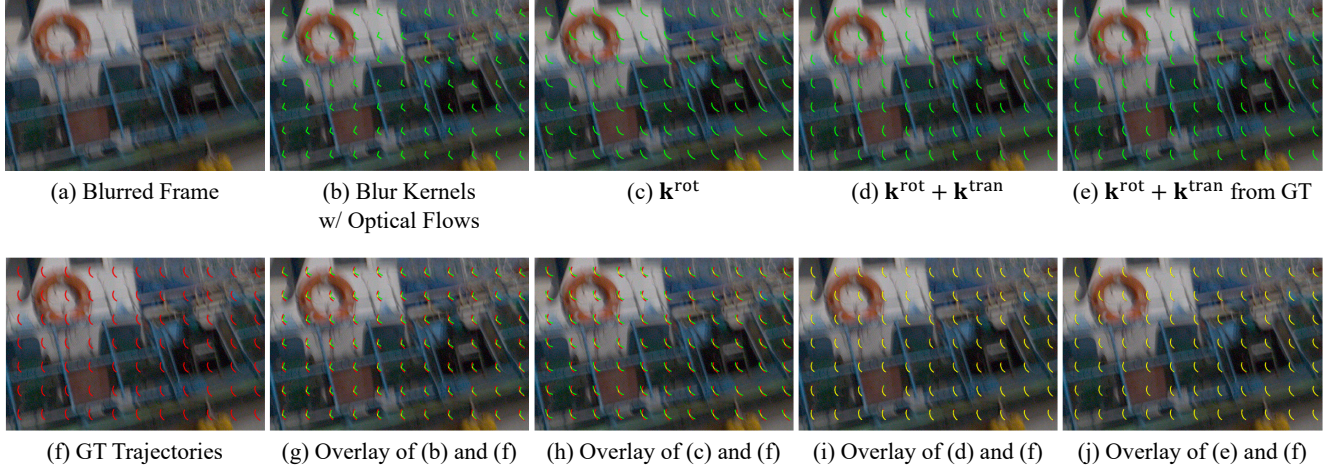


Figure S4. Visualization of blur kernels. (g)-(j) show the overlaid images, where blur kernels and GT trajectories are visualized in red and green, respectively. Yellow trajectories appear where the blur kernels and the GT trajectories overlap, indicating how well blur kernels are aligned with GT trajectories.

able because the true angular and translational velocities cannot be accessed. Instead, we use the trajectories computed from the source sharp frames before the averaging (*i.e.*, the frames used to generate the synthetic averaged images) as the ground-truth trajectories. Specifically, for each blurred frame, we estimate the optical flows between the source sharp frames and then compute pixel-wise trajectories by accumulating the optical flows from the center frame toward both the first and last frames. We use these trajectories as ground-truth motion trajectories. To match blur kernels with the ground-truth trajectories temporally, we construct blur kernels using the temporal centers of the source sharp frames as sample points, instead of  $\{t_i\}$ . Finally, we evaluate the quality of the blur kernels by measuring the mean of  $\ell_1$  distance error between the ground-truth trajectories and the computed blur trajectories.

Tab. S3 shows the quantitative evaluation of blur kernels. As shown in the table, blur kernels computed from optical flows suffer from large errors due to their limited temporal sampling. Also, rotation-only blur kernels (*i.e.*, w/  $\mathbf{k}^{\text{rot}}$  only) produce large errors because they ignore translational motion. In contrast, the errors decrease significantly when  $\mathbf{k}^{\text{tran}}$  is incorporated into the blur kernels. This result again shows that the translational motion component  $\mathbf{k}^{\text{tran}}$  is crucial for the quality of the final blur kernels. We estimate  $\mathbf{k}^{\text{tran}}$  using optical flows estimated from blurred video, which may contain errors and noise due to the presence of blur. As shown in the table, using ground-truth sharp frames to compute  $\mathbf{k}^{\text{tran}}$  yields the best accuracy. Nevertheless, the errors are reduced substantially even when  $\mathbf{k}^{\text{tran}}$  is estimated from blurred frames. Also, the resulting deblurring performance remains comparable, as demonstrated in Sec. 7.2. of the main paper.

Fig. S4 shows visualizations of the blur kernels. For ref-

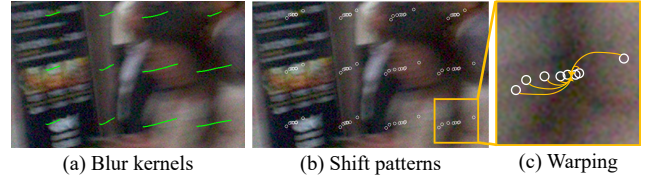


Figure S5. Visualization of blur kernels and the corresponding estimated shift patterns.

erence, we also include a visualization of the ground-truth trajectories and the overlaid images. The figure shows that blur kernels computed from optical flows capture only two-directional motion and therefore cannot represent complex motion (*e.g.*, curved motion). The rotation-only blur kernels capture a similar blur trajectory to the ground-truth trajectories but are misaligned because translational motion is ignored. In contrast, the proposed blur kernels that combine  $(\mathbf{k}^{\text{rot}}, \mathbf{k}^{\text{tran}})$  are well aligned with the ground-truth trajectories. Also, the figure shows that the resulting blur kernels are comparable in quality to those obtained when  $\mathbf{k}^{\text{tran}}$  is computed from ground-truth sharp frames.

## S8. Additional Metrics

In the main paper, we report only PSNR and SSIM for the evaluation due to space limitations. In this section, we additionally present further evaluation metrics. Tab. S4 presents a quantitative comparison using two perceptual quality metrics (*i.e.*, LPIPS [23] and DISTS [7]) and a temporal consistency metric (*i.e.*, tOF [6]). The table shows that GyroDVD also achieves superior performance in both perceptual quality and temporal consistency.



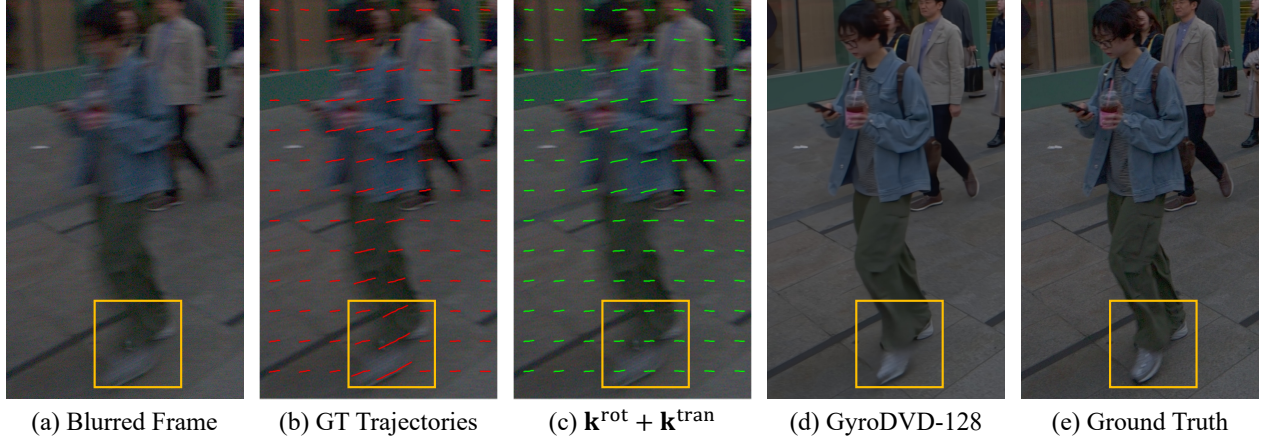


Figure S6. Visualization of blur kernels and the corresponding deblurred result in a failure case.

Table S4. Additional quantitative comparison on GyroVD-Syn. In addition to PSNR and SSIM, we report three additional metrics (*i.e.*, LPIPS [23], DISTS [7], and tOF [6]).

	GyroVD-Syn
	PSNR $\uparrow$ / SSIM $\uparrow$ / LPIPS $\downarrow$ / DISTS $\downarrow$ / tOF $\downarrow$
DeepGyro [13]	30.13 / 0.8003 / 0.334 / 0.165 / 11.24
EggNet [9]	30.32 / 0.8066 / 0.332 / 0.164 / 10.87
GyroDeblur [20]	32.34 / 0.8458 / 0.272 / 0.126 / 8.84
BasicVSR++ [5]	33.22 / 0.8689 / 0.231 / 0.102 / 2.97
EDVR [19]	33.31 / 0.8669 / 0.233 / 0.104 / 2.92
STCT [22]	33.37 / 0.8729 / 0.234 / 0.107 / 3.08
DSTNet [15]	33.86 / 0.8810 / 0.229 / 0.102 / 2.82
ShiftNet [10]	34.37 / 0.8865 / 0.221 / 0.102 / 2.50
VRT [12]	34.77 / 0.8948 / 0.204 / 0.094 / 2.54
RVRT [11]	34.82 / 0.8957 / 0.199 / 0.087 / 2.46
DSTNet+L [16]	34.90 / 0.8990 / 0.200 / 0.090 / 2.31
ShiftNet+ [10]	35.31 / 0.9023 / 0.195 / 0.089 / 2.29
ShiftNet [10] with $\mathbf{k}^{\text{rot}}$	34.55 / 0.8898 / 0.216 / 0.099 / 2.30
GyroDVD-48	35.12 / 0.9006 / 0.200 / 0.092 / 2.23
GyroDVD-64	35.39 / 0.9047 / 0.193 / 0.088 / 2.18
GyroDVD-96	35.76 / 0.9099 / 0.185 / 0.084 / 2.21
GyroDVD-128	35.93 / 0.9113 / 0.183 / 0.083 / 2.15

## S9. Visualization of KGS-Block

Fig. S5 visualizes the blur kernels and the estimated shift patterns in a KGS-Block. The figure shows estimated patterns preserve the underlying blur trajectories and closely align with the corresponding blur kernels. Specifically, the learned shift patterns effectively capture pixel-wise motion, enabling the network to focus on the blur kernel trajectories. By warping features according to these patterns, features along the blur trajectories are spatially aligned to the target pixel location, and subsequently propagated and aggregated across frames. This process enables spatially varying, motion-aware feature propagation, leading to more effective video deblurring.

## S10. Failure Case

GyroDVD relies on optical flow maps to compute the translational components of blur kernels, and its performance may degrade when optical flow estimation fails severely. To mitigate this issue, we employ separate kernel encoding layers to better handle errors in the blur kernels. In addition, we compute a consistency mask from the optical flows and mask unreliable regions where inconsistencies are detected. For the masked regions, we discard  $\mathbf{k}^{\text{tran}}$  and instead use those from nearest neighbors. Since blur kernels are generally spatially smooth, this design improves robustness to errors in optical flow estimation. However, when optical flow fails to capture object motion, especially for small objects, the resulting errors in the translational component remain difficult to handle.

Fig. S6 shows a deblurred result of moving objects along with the estimated blur kernels. In this example, the optical flow fails to capture the motion of the moving feet due to their small size. Consequently, the estimated translational component of the blur kernel becomes inaccurate, leading to artifacts of deblurring in those regions. Addressing these limitations remains an interesting direction for future work.

## S11. Additional Results

We present additional qualitative results on GyroVD-Syn (Fig. S7) and GyroVD-Real (Fig. S8).

## References

- [1] Android Developers. COLOR\_CORRECTION\_GAINS API. [https://developer.android.com/reference/android/hardware/camera2/CaptureResult#COLOR\\_CORRECTION\\_GAINS](https://developer.android.com/reference/android/hardware/camera2/CaptureResult#COLOR_CORRECTION_GAINS), . 5
- [2] Android Developers. COLOR\_CORRECTION\_TRANSFORM API. <https://developer.android.com/reference/android/hardware/camera2/>



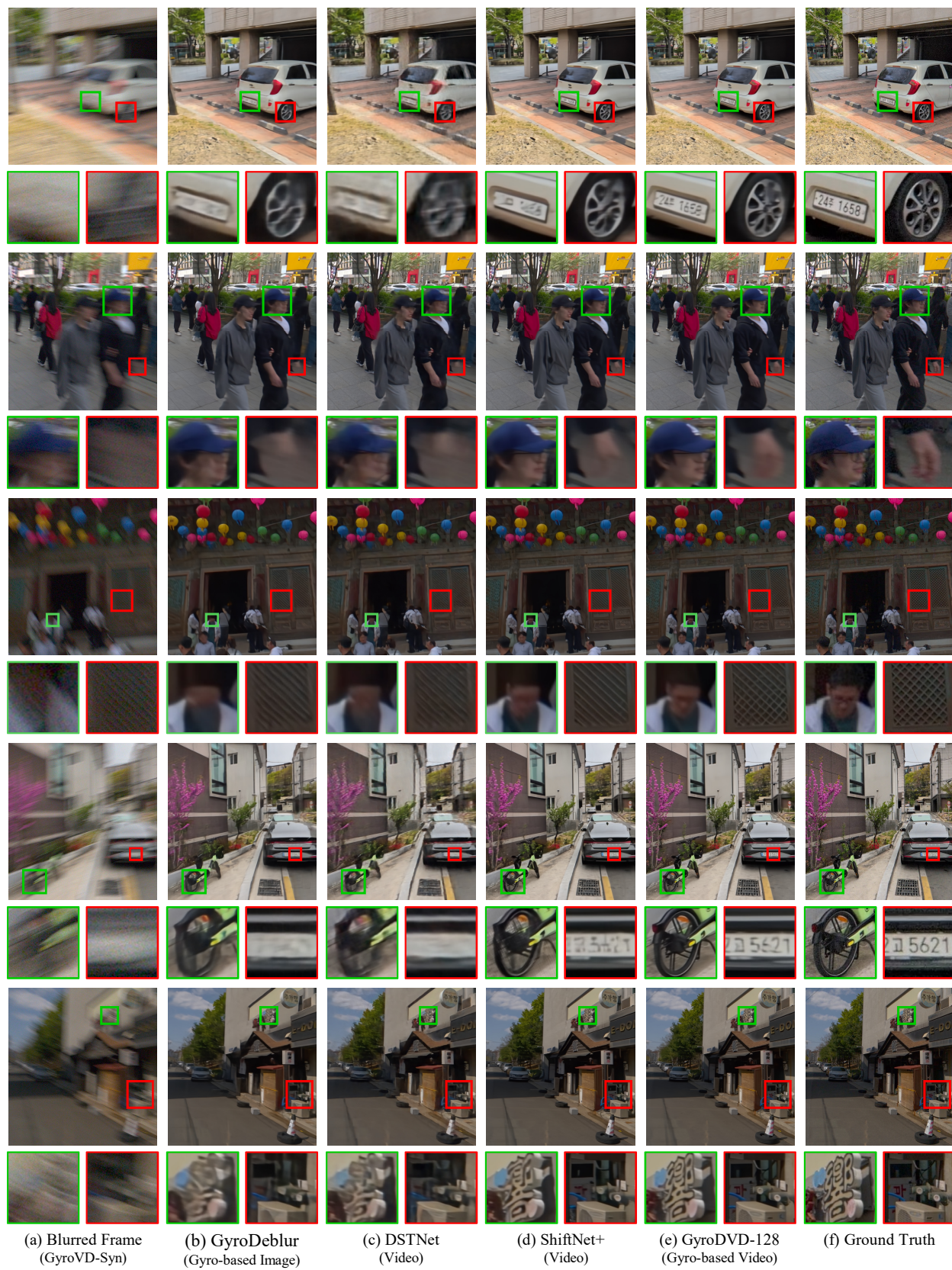


Figure S7. Qualitative results on GyroVD-Syn.





Figure S8. Qualitative results on GyroVD-Real.

`CaptureResult` # `COLOR` — `CORRECTION` — `TRANSFORM`, . 5

- [3] Android Developers. `SENSOR_NOISE_PROFILE` API. [https://developer.android.com/reference/android/hardware/camera2/CaptureResult#SENSOR\\_NOISE\\_PROFILE](https://developer.android.com/reference/android/hardware/camera2/CaptureResult#SENSOR_NOISE_PROFILE), . 6
- [4] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T Barron. Unprocessing images for learned raw denoising. In *Proc. of CVPR*, 2019. 5
- [5] Kelvin C.K. Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. BasicVSR++: Improving video super-resolution with enhanced propagation and alignment. In *Proc. of CVPR*, 2022. 4, 8
- [6] Mengyu Chu, You Xie, Jonas Mayer, Laura Leal-Taixé, and Nils Thuerey. Learning temporal coherence via self-supervision for gan-based video generation. *ACM Trans. Graph.*, 39(4):75–1, 2020. 7, 8
- [7] Keyan Ding, Kede Ma, Shiqi Wang, and Eero P Simoncelli. Image quality assessment: Unifying structure and texture similarity. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(5): 2567–2581, 2020. 7, 8
- [8] Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Toward convolutional blind denoising of real photographs. In *Proc. of CVPR*, 2019. 5
- [9] Seowon Ji, Jun-Pyo Hong, Jeongmin Lee, Seung-Jin Baek, and Sung-Jea Ko. Robust single image deblurring using gyroscope sensor. *IEEE Access*, 2021. 4, 8
- [10] Dasong Li, Xiaoyu Shi, Yi Zhang, Ka Chun Cheung, Simon See, Xiaogang Wang, Hongwei Qin, and Hongsheng Li. A simple baseline for video restoration with grouped spatial-temporal shift. In *Proc. of CVPR*, 2023. 4, 5, 8
- [11] Jingyun Liang, Yuchen Fan, Xiaoyu Xiang, Rakesh Ranjan, Eddy Ilg, Simon Green, Jiezhong Cao, Kai Zhang, Radu Timofte, and Luc V Gool. Recurrent video restoration transformer with guided deformable attention. In *Proc. of NeurIPS*, 2022. 4, 8
- [12] Jingyun Liang, Jiezhong Cao, Yuchen Fan, Kai Zhang, Rakesh Ranjan, Yawei Li, Radu Timofte, and Luc Van Gool. Vrt: A video restoration transformer. *IEEE Trans. Image Process.*, 33:2171–2182, 2024. 4, 8
- [13] Janne Mustaniemi, Juho Kannala, Simo Särkkä, Jiri Matas, and Janne Heikkilä. Gyroscope-aided motion deblurring with deep networks. In *Proc. WACV*, 2019. 4, 8
- [14] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *Proc. of CVPRW*, 2019. 5
- [15] Jinshan Pan, Boming Xu, Jiangxin Dong, Jianjun Ge, and Jinhui Tang. Deep discriminative spatial and temporal network for efficient video deblurring. In *Proc. of CVPR*, 2023. 4, 8
- [16] Jinshan Pan, Long Sun, Xu Boming, Jiangxin Dong, and Jinhui Tang. Learning efficient deep discriminative spatial and temporal networks for video deblurring. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2025. 4, 8
- [17] Jaesung Rim, Geonung Kim, Jungeon Kim, Junyong Lee, Seungyong Lee, and Sunghyun Cho. Realistic blur synthesis for learning image deblurring. In *Proc. of ECCV*, 2022. 5, 6
- [18] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *Proc. of CVPR*, 2017. 5
- [19] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *Proc. of CVPRW*, 2019. 4, 8
- [20] Heemin Yang, Jaesung Rim, Seungyong Lee, Seung-Hwan Baek, and Sunghyun Cho. Gyro-based neural single image deblurring. In *Proc. of CVPR*, 2025. 3, 4, 8
- [21] Guozhen Zhang, Yuhang Zhu, Haonan Wang, Youxin Chen, Gangshan Wu, and Limin Wang. Extracting motion and appearance via inter-frame attention for efficient video frame interpolation. In *Proc. of CVPR*, 2023. 5
- [22] Liyan Zhang, Boming Xu, Zhongbao Yang, and Jinshan Pan. Deblurring videos using spatial-temporal contextual transformer with feature propagation. *IEEE Trans. Image Process.*, 2024. 4, 8
- [23] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. of CVPR*, 2018. 7, 8
- [24] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proc. of CVPR*, 2019. 5