# Realistic Blur Synthesis for Learning Image Deblurring — Supplementary Material —

Jaesung Rim, Geonung Kim, Jungeon Kim, Junyong Lee,
Seungyong Lee, and Sunghyun Cho

POSTECH, Pohang, Korea
{jsrim123,k2woong92,jungeonkim,junyonglee,leesy,s.cho}@postech.ac.kr

## 1 Statistics of the RSBlur

Table 1 shows a statistical comparison with the RSBlur and other real-world blur datasets. The proposed RSBlur dataset consists of 13,358 real blurred images, which make the dataset the second largest real-world dataset. While the RealBlur [19] and BSD [25, 26] datasets consist of real blurred images and ground-truth sharp images, we provide real blurred images and sequences of nine sharp images to enable analysis on the blur generation process between real blurred and synthetic blurred images. In terms of image resolution, the RSBlur dataset provides the largest resolution. We also report the estimated noise levels using a single image noise estimation method [5] for comparing the amounts of noise in the real-world blurred datasets.

**Table 1.** Statistical comparison of real-world blur datasets. The average noise levels are estimated using a single image noise estimation method [5].

| | Frames | Real/Synth. | Resolution | Shutter (ms) | Noise |
|---|---|---|---|---|---|
| RealBlur [19] | 4,738 | Real | $680 \times 773$ | 500 | 0.4378 |
| BSD [25, 26] | 33,000 | Real | $640 \times 480$ | 8, 16, 24 | 0.3404 |
| RSBlur | 13,358 | Real & Synth | $1920 \times 1200$ | 100 | 0.7736 |

## 2 Real-world Deblurring Benchmark on the RSBlur

While there exist a couple of real-world blur datasets such as RealBlur [19] and BSD [25, 26], their coverage is limited. The real-world blurred images in the RSBlur dataset can also serve as an additional benchmark dataset that complements the existing benchmark datasets in terms of coverage.

In this section, we provide a benchmark on recent state-of-the-art deblurring methods using the RSBlur dataset to provide a basis for future deblurring research. Using real-world blurred images of the RSBlur dataset, we train

**Table 2.** Benchmark of state-of-the-art deblurring methods on the real blurred test set of the RSBlur dataset. We trained all methods using real blurred training set of the RSBlur dataset.

| Methods | PSNR / SSIM |
|---|---|
| SRN-Deblur [21] | 32.53 / 0.8398 |
| MiMO-UNet [7] | 32.73 / 0.8457 |
| MiMO-UNet+ [7] | 33.37 / 0.8560 |
| MPRNet [24] | 33.61 / 0.8614 |
| Restormer [23] | 33.69 / 0.8628 |
| Uformer-B [22] | 33.98 / 0.8660 |

state-of-the-art deblurring methods [21, 7, 24, 23, 22]. We use the source codes provided by the authors for training, and evaluate their performance using the real-blur test set of the RSBlur dataset. Here, we briefly report the qualitative and quantitative results of the state-of-the-art methods in Table 2 and in Fig. 2, respectively.

## 3 Details of Dual-Camera System

In this section, we describe the details of our dual-camera system. Fig. 2 shows our dual-camera system and a diagram of the system. The system consists of a mount for the lens, one beam splitter, and two camera modules with imaging sensors (Basler daA1920-160uc) so that the camera modules can capture the same scene while sharing one lens. For the lens, we used a Samyang 10mm F2.8 ED AS NCS CS. We installed a 5% neutral density filter (OD 1.3 VIS, 12.5mm Dia. Non-Reflective ND Filter) in front of a camera module. For compensation of beam-splitter tolerance, a 63% neutral density filter (0.2 OD, 25mm Dia., Precision Absorptive ND Filter) is installed in front of the other camera module. Two camera modules are installed on adjustable plates, so we can physically align the modules by adjusting the plates.

One camera module with a 5% neutral density filter captures a blurred image with a long exposure time (0.1 seconds). The other module captures nine sharp images with a short exposure time (0.005 seconds) during the exposure time of a blurred image. The gains of the two modules are set to 0 for both. To increase the number of images and diversity of blur, we capture 20 pairs of a blurred image and a sequence of nine sharp images of the same scene.

## 4 Camera ISP

To collect our dataset, we captured all images in the camera RAW format, and converted them into the nonlinear sRGB space using a simple image signal processing (ISP) pipeline. Our ISP consists of four steps: 1) white balance, 2) demosaicing, 3) color correction, and 4) conversion to the sRGB space using a camera response function. We use the demosaicing method of Malvar *et al.* [14]

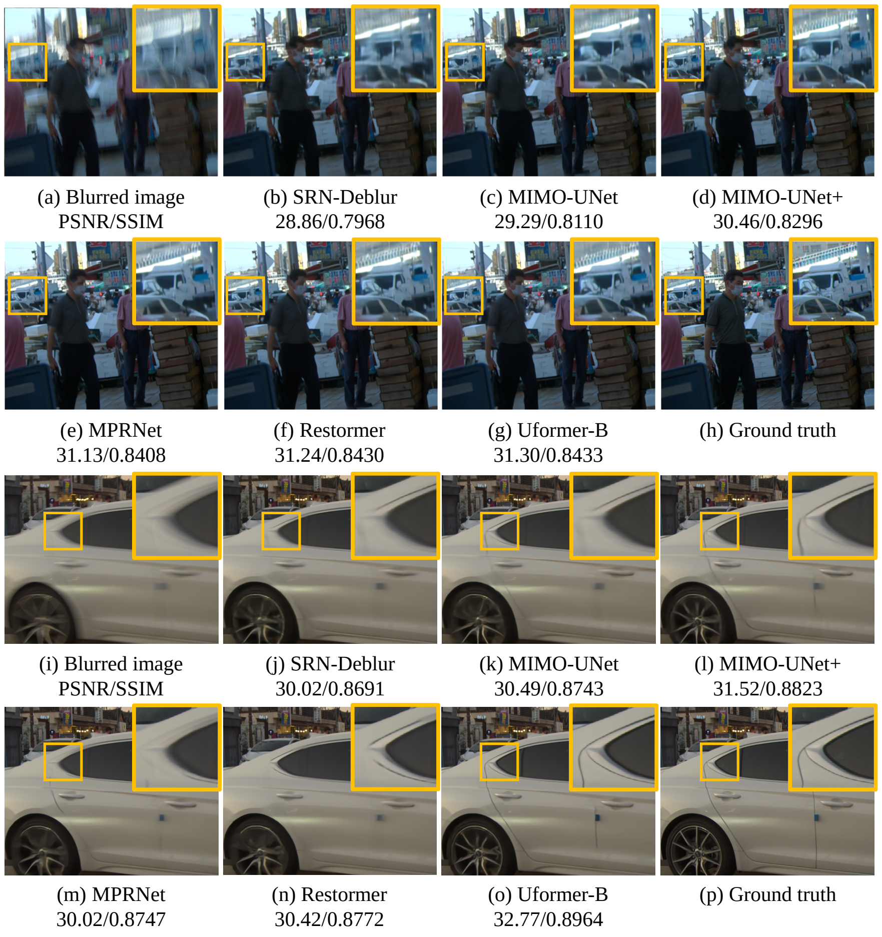


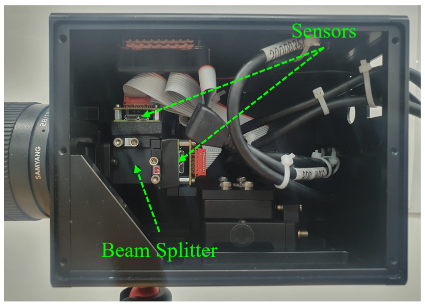
**Fig. 1.** Qualitative comparison of state-of-the-art deblurring methods on real-world blurred images of the RSBlur test set.

for the second step and a gamma correction of standard RGB space for the fourth step.

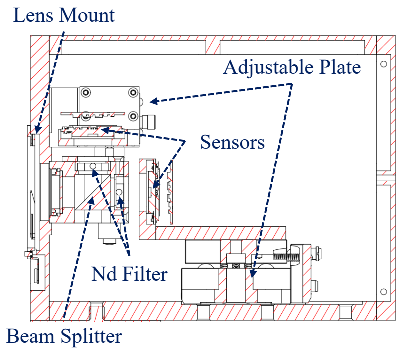
For the white balance and color correction steps in our ISP, we utilize a color chart. Specifically, when collecting our dataset, we captured reference images of a color chart for different scenes. Using the reference images, we estimate the gain $g_c$ for the color channel $c \in \{R, G, B\}$ for the white balance as:

$$g_c = \frac{P_n(G)}{P_n(c)} \tag{1}$$

where $P_n(c)$ is the mean intensity of the color channel $c$ of neutral patches in the color chart. Then, in the first step of our ISP, each color channel of a RAW image is multiplied by the corresponding gain.



(a) Our dual-camera system



(b) Diagram of our system

**Fig. 2.** The dual-camera system and detailed diagram of the system.

For the color correction in the third step of our ISP, we estimate a color correction matrix of the XYZ color space as:

$$\begin{bmatrix} X_1^{ref} & \cdots & X_{24}^{ref} \\ Y_1^{ref} & \cdots & Y_{24}^{ref} \\ Z_1^{ref} & \cdots & Z_{24}^{ref} \end{bmatrix} = \alpha \cdot T \begin{bmatrix} X_1 & \cdots & X_{24} \\ Y_1 & \cdots & Y_{24} \\ Z_1 & \cdots & Z_{24} \end{bmatrix} \tag{2}$$

where $(X_i^{ref}, Y_i^{ref}, Z_i^{ref})$ is the reference XYZ color of the $i$-th color chart patch, and $(X_i, Y_i, Z_i)$ is the measured XYZ color after white balancing and demosaicing. $\alpha$ is a single scalar value for matching the brightness levels of the color chart patches and the captured patches. $T$ is a $3 \times 3$ color correction matrix. We first estimate $\alpha$ by minimizing the mean-squared error between the color chart patches and captured patches. Then, we estimate $T$ by finding the least-squares solution of Eq. (2) with fixed $\alpha$.

Once a color correction matrix $T$ is obtained, we apply $T$ in the third step of our ISP as follows:

$$\begin{bmatrix} R_{Lin} \\ G_{Lin} \\ B_{Lin} \end{bmatrix} = M_{XYZ2Lin} \cdot T \cdot M_{Lin2XYZ} \cdot \begin{bmatrix} R_{Dem} \\ G_{Dem} \\ B_{Dem} \end{bmatrix} \tag{3}$$

where $(R_{Dem}, G_{Dem}, B_{Dem})$ is an RGB color of an image after demosaicing, and $(R_{Lin}, G_{Lin}, B_{Lin})$ is a resulting RGB color in the linear sRGB space. $M_{Lin2XYZ}$ and $M_{XYZ2Lin}$ are matrices for color conversion between the linear sRGB and XYZ color spaces. Fig. 3 shows intermediate results of our camera ISP.

## 5 Photometric Alignment between Camera Modules

Due to the optical spectrum difference caused by the beam splitter and ND filters, captured images may have slight photometric misalignments. To mitigate this, we conduct photometric alignment using a color chart image after the color correction step of our camera ISP. Specifically, we formulate the relationship between the colors of images from the two camera modules as:

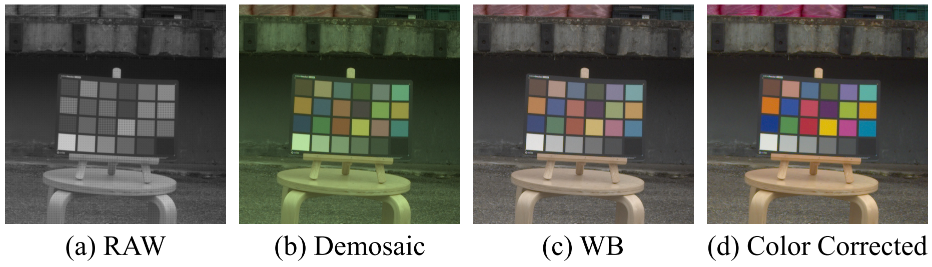


**Fig. 3.** Intermediate images of our camera ISP. All images are gamma corrected for visualization.

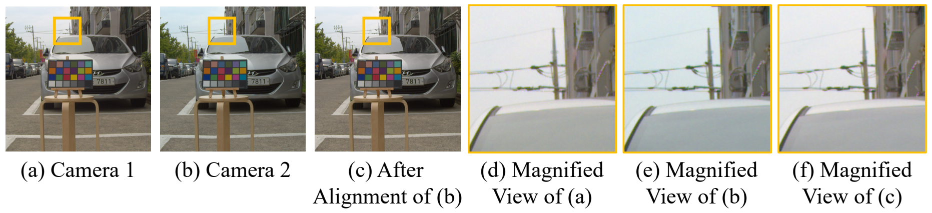

(a) Camera 1
(b) Camera 2
(c) After Alignment of (b)
(d) Magnified View of (a)
(e) Magnified View of (b)
(f) Magnified View of (c)

**Fig. 4.** Result of photometric alignment. (a) & (b) show captured images from a one camera module and the other camera module, respectively. (c) shows the photometric alignment result of (b). (d)-(f) show magnified views of (a)-(c).

$$\begin{bmatrix} X_1^{C1} & \cdots & X_{24}^{C1} \\ Y_1^{C1} & \cdots & Y_{24}^{C1} \\ Z_1^{C1} & \cdots & Z_{24}^{C1} \end{bmatrix} = T_p \begin{bmatrix} X_1^{C2} & \cdots & X_{24}^{C2} \\ Y_1^{C2} & \cdots & Y_{24}^{C2} \\ Z_1^{C2} & \cdots & Z_{24}^{C2} \end{bmatrix} \tag{4}$$

where $(X_i^{C1}, Y_i^{C1}, Z_i^{C1})$ and $(X_i^{C2}, Y_i^{C2}, Z_i^{C2})$ are the XYZ color values of the $i$-th color chart patch after the color correction of one camera module (C1) and the other camera module (C2), respectively. $T_p$ is a $3 \times 3$ matrix for the photometric alignment. We estimate $T_p$ by finding the least-squares solution of Eq. (4). As shown in Fig. 4(a)-(b), images captured by the two camera modules have color differences. After photometric alignment, the color difference is significantly reduced, as shown in Fig. 4(c).

## 6 Geometric Alignment between Camera Modules

Although the two camera modules are physically aligned as much as possible, there may exist a small amount of geometric misalignment between images from them (Fig. 5(c)). Thus, after capturing images, we conduct geometric alignment to compensate for this. The geometric alignment is performed for each pair of a blurred image and its corresponding sharp image sequence as the degree of misalignment can vary with respect to the distance between the camera system and scene.

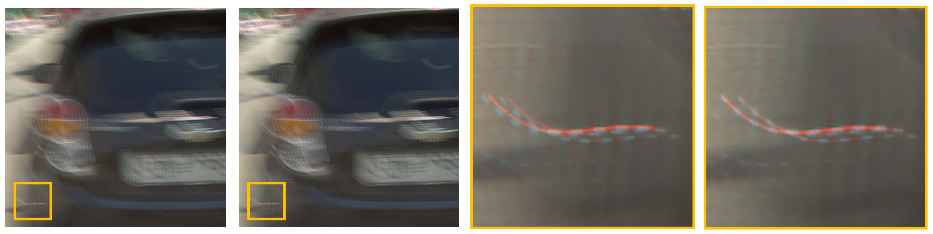

(a) Before Geometric Alignment (b) After Geometric Alignment (c) Magnified View of (a) (d) Magnified View of (b)

**Fig. 5.** Result of geometric alignment. (a) & (b) show stereo-anaglyph images, where a real blurred and the averaging of nine sharp images are visualized in red and cyan, respectively. (c) & (d) show magnified views of (a) and (b).

Specifically, for a given sharp image sequence, we first increase the frame rate 8× using a frame interpolation method [17], and synthesize a blurred image by averaging them. Then, we estimate a homography between a real blurred image and the synthesized one using the enhanced correlation coefficient method [8]. Finally, we warp the sharp images according to the estimated homography. We perform geometric alignment to the images processed by the ISP. Fig. 5(d) shows a result of our geometric alignment, where the red and cyan lights are better aligned after geometric alignment. Also, it shows that the real blurred image and nine sharp images are well synchronized.

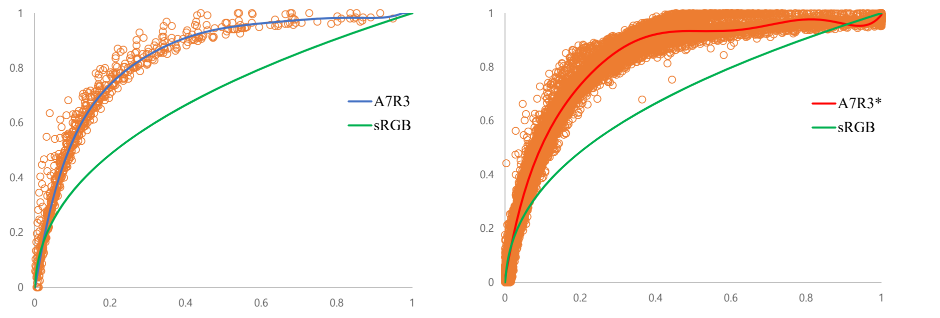
## 7 Camera ISP for the RealBlur Dataset

In the main paper, we improve the performance of SRN-DeblurNet [21] by mimicking the ISP of the Sony A7R3 camera, which is used for the RealBlur dataset [19]. Our camera ISP for the Sony A7R3 also consists of the white balance, demosaicing, color correction, and camera response function (CRF) steps.

As the ISP affects the noise distribution and non-linearity of the blur, we match the white balance gains, color correction matrix, and CRF as much as possible to those of the RealBlur dataset. For white balance, we extract the white balance gains from the RAW images of the RealBlur training set. Similar to [3], we randomly sample the gains from the RealBlur training set and multiply a RAW image by the gains. After white balancing, we apply the demosaicing method of Malvar *et al.* [14].

For the color correction, we extract a characterization matrix of A7R3 from the Libraw library and convert it into a color correction matrix following [20]. The extracted color correction matrix directly maps from the RAW space to the XYZ color space. We convert a demosaiced image into the linear sRGB space as:

$$\begin{bmatrix} R_{Lin} \\ G_{Lin} \\ B_{Lin} \end{bmatrix} = M_{XYZ2Lin} \cdot T_{A7R3} \cdot \begin{bmatrix} R_{Dem} \\ G_{Dem} \\ B_{Dem} \end{bmatrix} \quad (5)$$



(a) Estimated CRF using color chart (b) Estimated CRF using an image from the internet

**Fig. 6.** The blue and red lines show the estimated CRFs of the Sony A7R3 using color chart images and a raw-RGB image from the internet, respectively. The orange dots show measured values. The green lines show gamma correction of sRGB space.

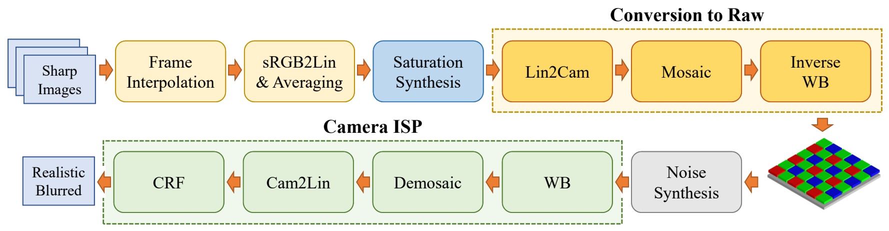


**Fig. 7.** Overview of our realistic blur synthesis pipeline. Lin2Cam: Inverse color correction, i.e., color space conversion from the linear sRGB space to the camera RAW space. WB: White balance. Cam2Lin: Color correction.

where $(R_{Dem}, G_{Dem}, B_{Dem})$ is an RGB color of an image after demosaicing, and $(R_{Lin}, G_{Lin}, B_{Lin})$ is a resulting RGB color in the linear sRGB space.

The final step applies a CRF. Motivated by [15], we model the CRF as a high-order polynomial as follows:

$$I_{srgb} = \sum_{k=0}^{K} c_k I_{Lin}^k \tag{6}$$

where $I_{srgb}$, $I_{Lin}$, and $c_k$ are a non-linear and a linear sRGB image, and polynomial coefficients, respectively. $K$ is the polynomial order, which is set to 7. We capture 11 color chart images with different shutter speeds using a Sony A7R3 camera. Then, we measure the RGB values of the color patches in JPEG images from the camera and in linear sRGB images converted from RAW images. Using the pairs of the measured RGB values, we estimate the coefficients of the CRF $c_k$ by solving a least-squares problem. Fig. 6(a) shows the estimated CRF using color chart images.

## 8 Conversion from sRGB to RAW

Fig. 7 shows an overview of our realistic blur synthesis pipeline. As described in the main paper, we convert the image from the saturation synthesis step into the mosaiced camera RAW space. To this end, we apply inverse color correction, mosaicing, and inverse white balance sequentially. Then, we apply the camera ISP to reflect distortions introduced by the camera ISP. In the case of Sony A7R3, we apply the ISP described in Sec. 7. In the following, we describe each step of the conversion to RAW in more detail.

**Lin2Cam** This step performs inverse color correction. As color correction is a simple linear operation, we can apply inverse color correction as follows:

$$\begin{bmatrix} R_{Cam} \\ G_{Cam} \\ B_{Cam} \end{bmatrix} = M_{XYZ2Lin} \cdot T^{-1} \cdot M_{Lin2XYZ} \cdot \begin{bmatrix} R_{Lin} \\ G_{Lin} \\ B_{Lin} \end{bmatrix} \tag{7}$$

where $(R_{Lin}, G_{Lin}, B_{Lin})$ and $(R_{Cam}, G_{Cam}, B_{Cam})$ are RGB colors in the linear sRGB and RAW RGB spaces, respectively. $M_{Lin2XYZ}$ and $M_{XYZ2Lin}$ are matrices for color conversion between the linear sRGB and XYZ color spaces. $T^{-1}$ is the inverse of a color correction matrix $T$.

In the case of the Sony A7R3, the color correction matrix $T_{A7R3}$ directly maps colors in the RAW color space into the XYZ color space as mentioned in Sec. 7. Thus, we perform inverse color correction as:

$$\begin{bmatrix} R_{Cam} \\ G_{Cam} \\ B_{Cam} \end{bmatrix} = T^{-1}_{A7R3} \cdot M_{Lin2XYZ} \cdot \begin{bmatrix} R_{Lin} \\ G_{Lin} \\ B_{Lin} \end{bmatrix} \tag{8}$$

where $(R_{Lin}, G_{Lin}, B_{Lin})$ and $(R_{Cam}, G_{Cam}, B_{Cam})$ are RGB colors in the linear sRGB and RAW RGB spaces, respectively.

**Mosaic** Following [9], we randomly sample a Bayer pattern from RGGB, BGGR, GRBG, and GBRG to reflect distortions caused by various Bayer patterns. Then, we perform mosaicing using the sampled pattern.

**Inverse WB** As we already know white balance gains for each image, in this step, we simply apply their inverse $g_c^{-1}$ to each color channel of a mosaiced image.

## 9 Additional Analysis Results

In this section, we also provide additional analysis results using MIMO-Unet [7]. For the experiments, we train MIMO-Unet model for 790K iterations, which is half the number of iterations suggested in [7], with variants of our pipeline. Table 3 shows that the method 2 performs worse than method 1, which uses real blurred images for training. Methods 3 and 4 show that adding Gaussian noise ($\sigma = 0.0112$) and our saturation synthesis significantly improves the deblurring performance. The method 5, which corresponds to our full pipeline, achieves

**Table 3.** Additional analysis using MIMO-Unet [7] on the RSBlur dataset. Interp.: Frame interpolation. Sat.: Saturation synthesis. sRGB: Gamma correction of sRGB space. G: Gaussian noise. G+P: Gaussian and Poisson noise.

| Blur Synthesis Methods | | | | | | | PSNR / SSIM | | |
|---|---|---|---|---|---|---|---|---|---|
| No. | Real | CRF | Interp. | Sat. | Noise | ISP | All | Saturated | No Saturated |
| 1 | ✓ | | | | | | 32.73 / 0.8457 | 31.44 / 0.8385 | 33.93 / 0.8524 |
| 2 | | sRGB | ✓ | | | | 28.83 / 0.7164 | 27.42 / 0.7052 | 30.16 / 0.7270 |
| 3 | | sRGB | ✓ | | G | | 29.63 / 0.7552 | 28.28 / 0.7486 | 30.90 / 0.7614 |
| 4 | | sRGB | ✓ | Ours | G | | 29.84 / 0.7658 | 28.49 / 0.7590 | 31.12 / 0.7723 |
| 5 | | sRGB | ✓ | Ours | G+P | ✓ | 32.08 / 0.8362 | 30.68 / 0.8290 | 33.39 / 0.8429 |

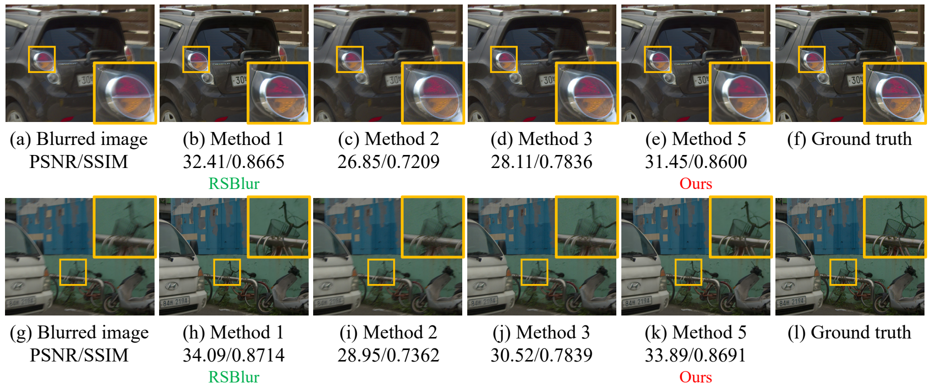


**Fig. 8.** Qualitative comparison of deblurring results on the RSBlur test set produced by MIMO-Unet [7] trained with different synthesis methods. (b)-(e) & (h)-(k) Methods 1, 2, 3 and 5 in Table 3. Best viewed in zoom in.

32.08 dB. The analysis shows that MIMO-Unet [7] also has significant performance improvement with the proposed synthesis pipeline. Fig. 8 shows results of MIMO-Unet [7] trained on the RSBlur dataset with different methods in Table 3.

## 10 Experiments with Rough Camera Parameters

Estimating accurate camera-specific parameters can be easily done by taking a few shots of images. Even if the camera is not available, rough parameters can be easily obtained using images available on the internet in the case of most consumer cameras. To show this, assuming that an A7R3 camera is not available, we conducted additional experiments on the RealBlur_J [19] dataset where we estimated camera-specific parameters from images from the internet.

In the case of most consumer cameras, the characterization matrix is easily obtained from the Libraw library[1]. As described in Sec. 7, we extract the characterization matrix of the A7R3 camera and convert it into a color correction matrix. The SIDD dataset [2] provides the noise parameters of four cameras on different ISO settings. As the RealBlur_J dataset is mostly captured with ISO

[1] https://github.com/LibRaw/src/tables/colordata.cpp

**Table 4.** Performance comparison of different blur synthesis methods on the RealBlur_J [19] test sets. Interp.: Frame interpolation. Sat.: Saturation synthesis. sRGB: Gamma correction of sRGB space. G: Gaussian noise. G+P: Gaussian and Poisson noise. A7R3: Using camera ISP with accurate parameters estimated from a Sony A7R3 camera. A7R3*: Using camera ISP with rough parameters.

| Blur Synthesis Methods | | | | | | | PSNR / SSIM |
|---|---|---|---|---|---|---|---|
| No. | Training set | CRF | Interp. | Sat. | Noise | ISP | RealBlur_J |
| 1 | RealBlur_J | | | | | | 30.79 / 0.8985 |
| 2 | BSD_All | | | | | | 28.66 / 0.8589 |
| 3 | GoPro | sRGB | ✓ | | | | 28.92 / 0.8711 |
| 4 | GoPro | sRGB, A7R3 | ✓ | Ours | G+P | A7R3 | 30.32 / 0.8899 |
| 5 | GoPro | sRGB, A7R3* | ✓ | Ours | G+P | A7R3* | 30.23 / 0.8864 |
| 6 | GoPro_U | sRGB | | | | | 29.28 / 0.8766 |
| 7 | GoPro_U | sRGB, A7R3 | | Ours | G+P | A7R3 | 30.75 / 0.9019 |
| 8 | GoPro_U | sRGB, A7R3* | | Ours | G+P | A7R3* | 30.55 / 0.8956 |

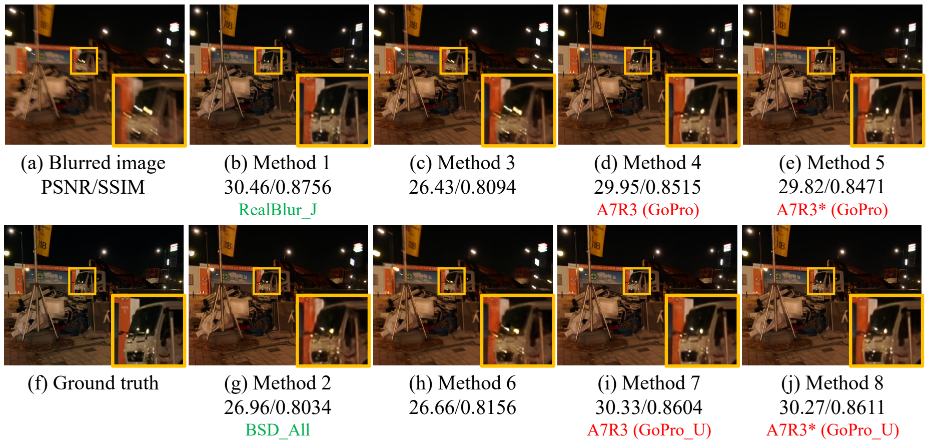


**Fig. 9.** Qualitative comparison of deblurring results on the RealBlur_J test set produced by models trained with different synthesis methods. (b)-(e) Methods 1, 3, 4 and 5 in Table 4. (g)-(j) Methods 2, 6, 7 and 8 in Table 4. Best viewed in zoom in.

100, we sample the noise parameters of Google Pixel on the ISO 100 setting. Following [3], we randomly sample gains for the red and blue channels from $\mathcal{U}(1.9, 2.4)$ and $\mathcal{U}(1.5, 1.9)$ for the white balance.

In the case of the CRF, there is no dataset including the CRFs of the latest consumer cameras, and it is difficult to estimate CRFs without cameras. Instead, we utilize the camera profile of Sony A7R3 available on Adobe Lightroom. Specifically, we first download a RAW image captured by an A7R3 from the internet. Then, we convert the RAW image into an sRGB image using the camera profile of Adobe Lightroom. We also convert the RAW RGB image into a linear sRGB image using the color correction matrix from the Libraw library and white balance gains of the RAW RGB image. Then, using the pixel values of the converted linear sRGB image and sRGB image, we estimate the coefficients of Eq. (6) by solving a least-squares problem. Fig. 6(b) shows the estimated

**Table 5.** Performance comparison of SRN-DeblurNet [21] trained on the RealBlur_J [19], BSD_All [25, 26], and RSBlur datasets.

| Test / Train | PSNR / SSIM | | |
|---|---|---|---|
| | RealBlur_J | BSD_All | RSBlur |
| RealBlur_J | 30.79 / 0.8985 | 29.67 / 0.8922 | 29.86 / 0.7895 |
| BSD_All | 28.66 / 0.8589 | 33.35 / 0.9348 | 30.89 / 0.8049 |
| RSBlur | 29.86 / 0.8855 | 30.85 / 0.9069 | 32.53 / 0.8398 |

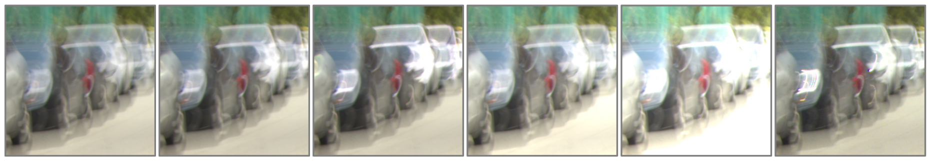

(a) Averaging (b) $B_{local}$ ($\alpha_1 = 1.5$) (c) $B_{local}$ ($\alpha_1 = 2.5$) (d) $B_{global}$ ($\alpha_1 = 1.5$) (e) $B_{global}$ ($\alpha_1 = 2.5$) (f) Real

**Fig. 10.** Generated images from saturation synthesis methods using global scaling and local scaling.

CRF using the RAW image resembles the estimated CRF using a color chart very closely.

To verify the effectiveness of the rough parameters estimated as described above, we train SRN-DeblurNet [21] using the proposed synthesis pipeline with the estimated parameters, and evaluate its performance. In Table 4, methods 5 and 8 that use the rough parameters perform worse than methods 4 and 7 that use accurate camera parameters. Neverthelss, compared to the methods 3 and 6, the methods 5 and 8 still perform significantly better, validating the effectiveness of the rough parameters. Fig. 9 shows qualitative examples of using the proposed pipeline with rough and accurate camera parameters and other naïve methods.

## 11 Limited Coverage of Real Datasets

In the main paper, we show the limited coverage of the existing real datasets including RealBlur [19] and BSD_All [25, 26]. Specifically, we train SRN-DeblurNet [21] using one dataset, and evaluate its performance on the other dataset. In this supplementary material, we report a full comparison result among the real datasets including the RSBlur real dataset in Table 5. As the table shows, the deblurring performance of one dataset significantly drops on the other datasets. This result again verifies the limited coverage of the existing real datasets and the usefulness of our synthesis pipeline.

## 12 Saturation Synthesis: Local vs. Global

One important component in our blur synthesis pipeline is the saturation synthesis step, which locally scales intensity values of a blurred image before clipping as done in [10]. Another option that has been used in several previous works is global scaling [16, 6, 18]. In this section, we discuss the global and local scaling approaches and compare their performance.

Without considering noise and camera ISP for the brevity of the discussion, we can model a blurred image $B$ with clipped intensity values as $B = \text{clip}[I * K]$, where $\text{clip}[\cdot]$ is a clipping function, $K$ is a convolution kernel, and $I$ is a sharp image. To obtain clipped intensity values in $B$, $I$ should contain unclipped intensity values larger than the upper limit of the dynamic range. However, as sharp images also have the limited dynamic range, it is inevitable to synthesize sharp images with large intensity values, which can be done by either global scaling [16, 6, 18] or local scaling [10].

The global scaling approach generates saturated pixels by scaling all the intensity values as $\text{clip}[\alpha_1 \cdot I * K]$, where $\alpha_1$ is a scaling factor. The local scaling approach, on the other hand, scales only some intensity values as $\text{clip}[(I^{n\text{-}sat} + \alpha_1 \cdot I^{sat}) * K]$, where $I^{n\text{-}sat}$ and $I^{sat}$ are images that have non-zero pixels on non-saturated and saturated region of $I$, respectively. Due to the information loss at clipped pixels, both methods randomly sample $\alpha_1$ to generate non-clipped pixels. By replacing the convolution operation with $K$ by averaging operation over consecutive video frames, we can also model the blur synthesis based on averaging video frames, on which our analysis in the main paper is based.

Fig. 10 shows real saturated images and synthesized images using global scaling and local scaling. Both methods cannot exactly reproduce the real saturated pixels (Fig. 10(c),(e), and (f)) due to the missing information in sharp images caused by clipping. Nevertheless, the local scaling approach has a couple of advantages over the global scaling approach. First, global scaling affects all the pixels, thus introduces a larger domain gap (brighter images) as shown in Fig. 10(e), and severe distortion of the distribution of signal-dependent noise. Second, as global scaling increases the intensities of larger areas, it is usually more difficult to mimic sharp light streaks often observed in real blurred images. This difference can also affect the deblurring performance. We conducted an experiment using the GoPro_U training set with global scaling, and found that global scaling performs worse than our saturation synthesis by 0.35 dB on the RealBlur test set.

## 13 Direct Measurement of the Quality of Blur Synthesis

To evaluate the synthesis methods, we measure the deblurring performance trained with them in the main paper. Another possible option would be to directly compare synthetic blurred images against real blurred images to quantify the quality of the synthetic methods using the RSBlur dataset. In this section, we discuss about the direct measurement of the quality of blur synthesis and report additional evaluation results.

To measure the quality of different blur synthesis approaches, we measure the PSNR and SSIM values of synthetic blurred images against real-blurred images. However, the PSNR or SSIM values of synthetic images are not 100% reliable due

**Table 6.** Comparison among different blur synthesis methods. We compute PSNR, SSIM and KL-divergence (KLD) from synthesized and real blurred images.

| Blur Synthesis Methods | | | | | |
|---|---|---|---|---|---|
| No. | Interp. | Sat. | Noise | ISP | PSNR / SSIM / KLD |
| 1 | | | | | 35.74 / 0.8802 / 1.0064 |
| 2 | ✓ | | | | 35.97 / 0.8888 / - |
| 3 | ✓ | Ours | | | 36.03 / 0.8888 / - |
| 4 | ✓ | | G | | 34.16 / 0.8075 / 0.4312 |
| 5 | ✓ | | G+P | ✓ | 33.14 / 0.7928 / 0.3319 |
| 6 | ✓ | Ours | G | | 34.21 / 0.8077 / 0.4313 |
| 7 | ✓ | Ours | G+P | ✓ | 33.18 / 0.7928 / 0.3263 |

to noise. One possible option is to measure the KL-divergence of the distributions of real and synthesized images as done in noise-synthesis approaches [1, 4, 11]. Specifically, we compute the KL-divergence between pixel-wise marginal distributions $(B_{real} - B_{interp})$ and $(B_{syn} - B_{interp})$ where $B_{real}$ and $B_{interp}$ is a real blurred image and an averaging of interpolated images, respectively. $B_{syn}$ is a synthesized blurred image using our pipeline. We use $B_{interp}$ as rough noise-free counterparts of real blurred images for computing the KL-divergence.

Table 6 shows PSNR, SSIM, and the KL-divergence of synthesized images using variants of our synthesis methods. The table shows that frame interpolation and saturation synthesis are effective in terms of PSNR (methods 2 and 3). The method 7 shows noise and saturation synthesis methods improve the KL-divergence. We omit the KD-divergence values of methods 2 and 3 as the synthetic images have no noise thus their noise distributions are not properly defined. Note that, measuring the KL-divergence also has its own flaws as we don't have accurate noise-free counterparts of real blurred images. So, 100% accurate estimation of real distribution is not feasible. Also, as our ultimate goal is to improve the deblurring quality, the deblurring performance of the trained network is a most reasonable measure.

## 14 Training with Both Real and Synthetic Datasets

Even if a real-world blur training set is available, an additional synthetic dataset generated by our method could further improve the deblurring performance on real blurred images. One important question when training with two datasets is how to mix them. To verify the effect of using both real and synthetic datasets and the mixing strategy, in this section, we compare the deblurring performance obtained using one of real and synthetic datasets and using both of them. Specifically, we compare four different training strategies: training 1) with only RealBlur_J, 2) with only GoPro_U, 3) with both RealBlur_J and GoPro_U together half and half at each iteration, and 4) with RealBlur_J and GoPro_U one by one at each iteration. We train SRN-DeblurNet using the four strategies and evaluate their performances on the RealBlur test set. As the table shows, using both

**Table 7.** Comparison of different training strategies using additional synthetic datasets for further performance improvements.

| No. | Train dataset | Strategy | PSNR / SSIM |
|---|---|---|---|
| 1 | RealBlur_J | | 30.79 / 0.8985 |
| 2 | GoPro_U | | 30.75 / 0.9019 |
| 3 | RealBlur_J + GoPro_U | Half & Half | 31.17 / 0.9065 |
| 4 | RealBlur_J + GoPro_U | One × One | 31.15 / 0.9059 |

RealBlur_J and GoPro_U clearly improves the deblurring performance regardless of the training strategies.

## 15 Additional Results on Other Datasets

In the main manuscript, we evaluate the proposed pipeline on the RealBlur [19] and BSD_All [25, 26] datasets. Additionally, in this supplementary material, we also report evaluation results on Köhler *et al.*'s dataset [12]. Table 8 shows the performance of SRN-DeblurNet [21] trained with variants of the proposed convolution-based synthesis pipeline. Note that the images in Köhler *et al.*'s dataset are in the linear RGB space, and do not have saturated pixels. Thus, the method 5 performs the best, while the other methods with wrong CRFs show significant performance drops. Again, the poor performances of the existing real datasets on Köhler *et al.*'s dataset prove their limited coverage, as discussed in the main manuscript.

**Table 8.** Performance comparison of different blur synthesis methods on the Köhler *et al.*'s [12] dataset. Sat.: Saturation synthesis. sRGB: Gamma correction of sRGB space. G: Gaussian noise. G+P: Gaussian and Poisson noise. A7R3: Using camera ISP parameters estimated from a Sony A7R3 camera, which was used for collecting the RealBlur dataset.

| Blur Synthesis Methods | | | | | | PSNR / MSSIM |
|---|---|---|---|---|---|---|
| No. | Training set | CRF | Sat. | Noise | ISP | Köhler *et al.* |
| 1 | RealBlur_J | | | | | 26.79 / 0.8401 |
| 2 | BSD_All | | | | | 25.24 / 0.7920 |
| 3 | RSBlur | | | | | 26.29 / 0.8285 |
| 4 | GoPro_U | Linear | | | | 28.28 / 0.8599 |
| 5 | GoPro_U | Linear | | G | | 28.41 / 0.8624 |
| 6 | GoPro_U | sRGB | | | | 26.86 / 0.8430 |
| 7 | GoPro_U | sRGB | | G | | 27.23 / 0.8529 |
| 8 | GoPro_U | sRGB | Ours | G | | 27.10 / 0.8500 |
| 9 | GoPro_U | sRGB, A7R3 | Ours | G+P | A7R3 | 27.41 / 0.8516 |

We also compare variants of the convolution-based synthesis pipeline in Table 8 on Lai *et al.*'s dataset [13], which provides 100 real blurred images without ground-truth sharp images for qualitative comparison. In Figs. 11 and 12, methods 1, 2, and 3 show that the models trained on real datasets do not successfully deblur real blurred images of Lai *et al.*'s dataset. On the other hand, the method 9 shows that the deblurring results produced by the model trained with our pipeline performs better. This again shows the limited coverage of the real datasets and the practicality of the proposed pipeline. The proposed pipeline can generate a huge number of images with various contents, and blur shapes and sizes effortlessly compared to collecting real datasets, leading to better deblurring performance.

## 16 Additional Qualitative Examples

Figs. 13, 14 and 15 show additional qualitative examples on the RSBlur, RealBlur_J [19], and BSD_All [25, 26] datasets, respectively. The figures show deblurring results of SRN-DeblurNet [21] trained with training images synthesized by different methods in order to compare different blur synthesis methods.

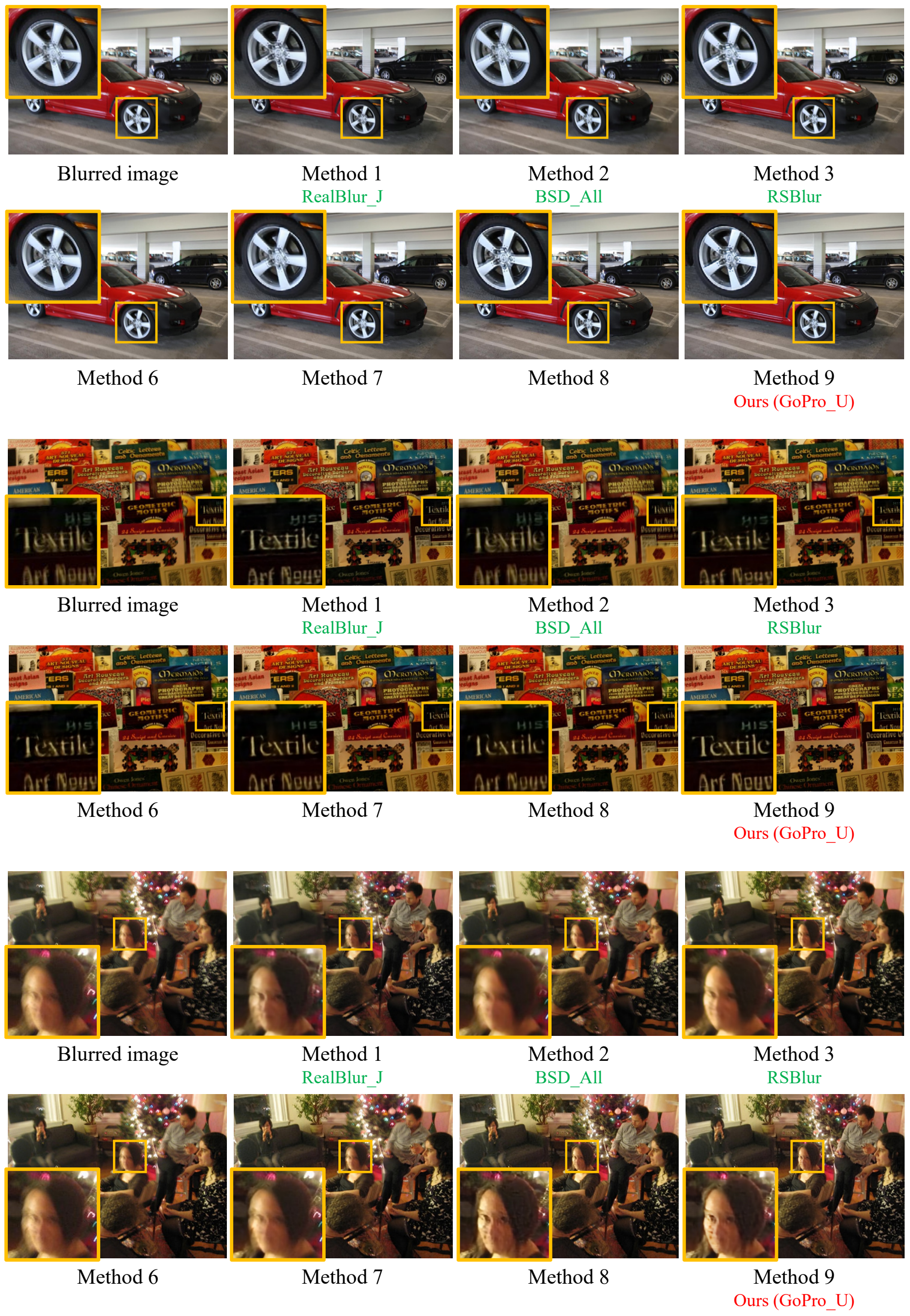


**Fig. 11.** Qualitative comparison of deblurring results on the Lai *et al.*'s dataset [13] produced by models trained with different synthesis methods in Table 8. Best viewed in zoom in.

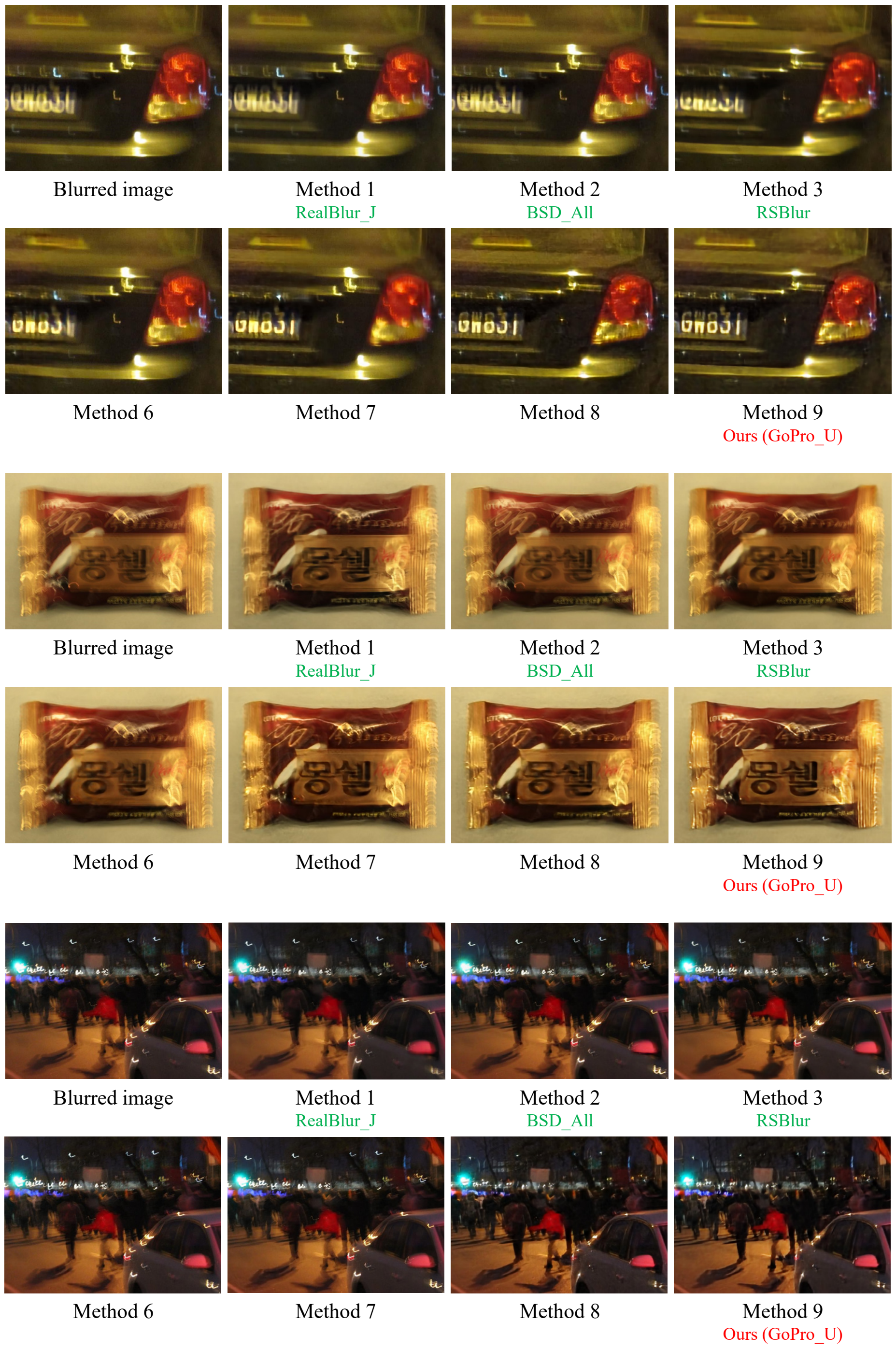


**Fig. 12.** Qualitative comparison of deblurring results on the Lai *et al.*'s dataset [13] produced by models trained with different synthesis methods in Table 8. Best viewed in zoom in.

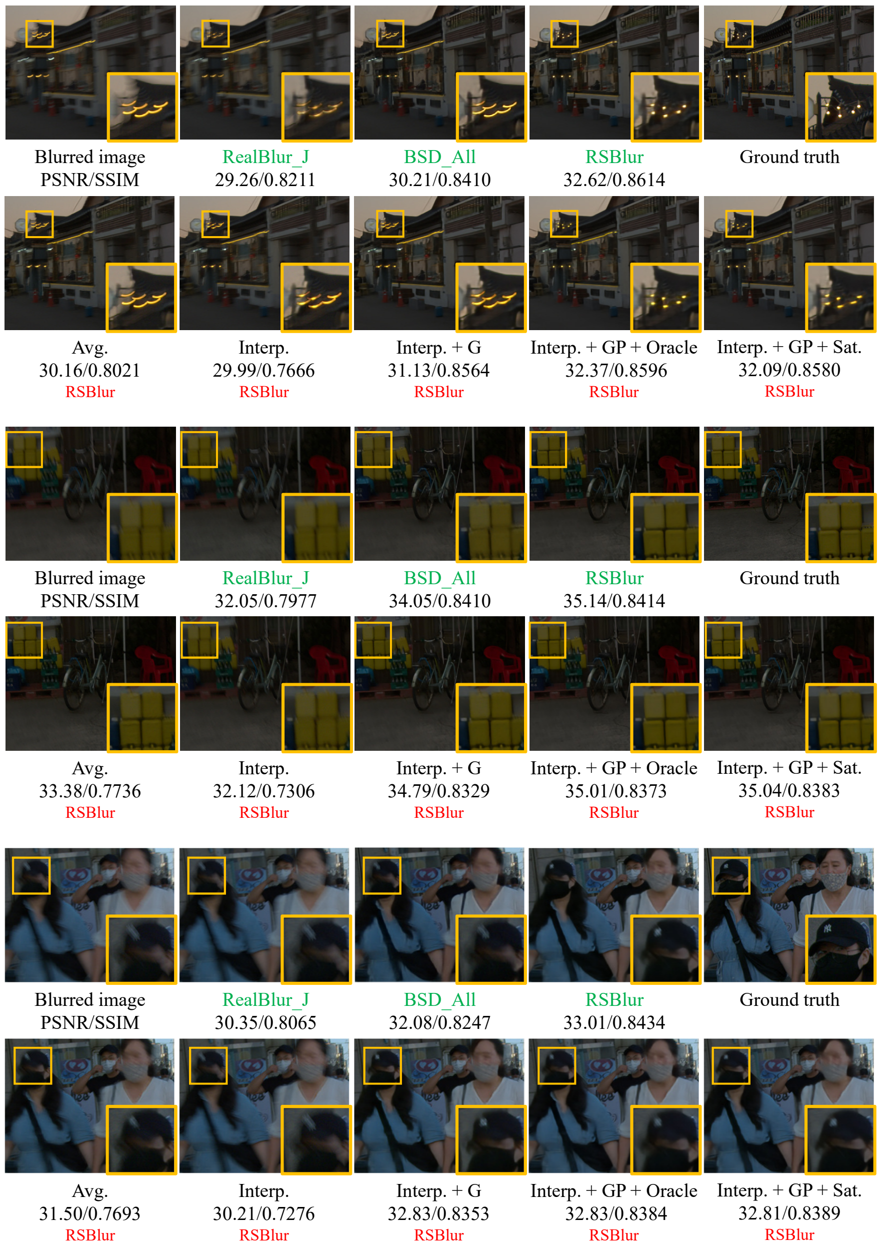


**Fig. 13.** Qualitative comparison on the RSBlur dataset. Green: Trained on real blurred images. Red: Trained on synthetic blurred images. Avg.: Naïve averaging-based blur synthesis. Interp.: Averaging-based blur synthesis using frame interpolation. G: Gaussian noise. Oracle: Using oracle saturated images. Sat: Our saturation synthesis. All of synthesis methods consider gamma decoding and encoding.

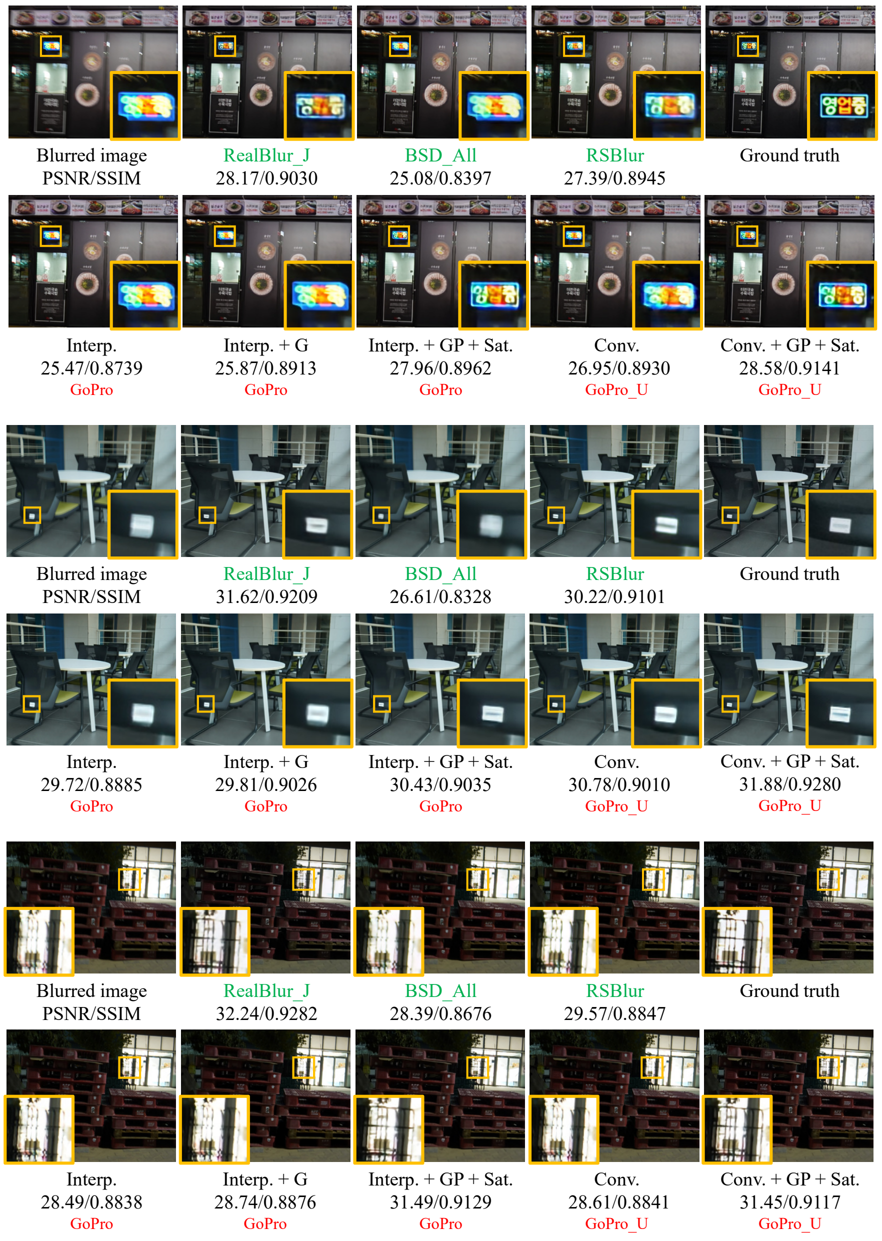


**Fig. 14.** Qualitative comparison on the RealBlur_J dataset [19]. Green: Trained on real blurred images. Red: Trained on synthetic blurred images. Interp.: Averaging-based blur synthesis using frame interpolation. G: Gaussian noise. GP: Gaussian and Poisson noise with a camera ISP of Sony A7R3. Sat: Our saturation synthesis. Conv.: Convolution-based blur synthesis. All of synthesis methods consider gamma decoding and encoding.

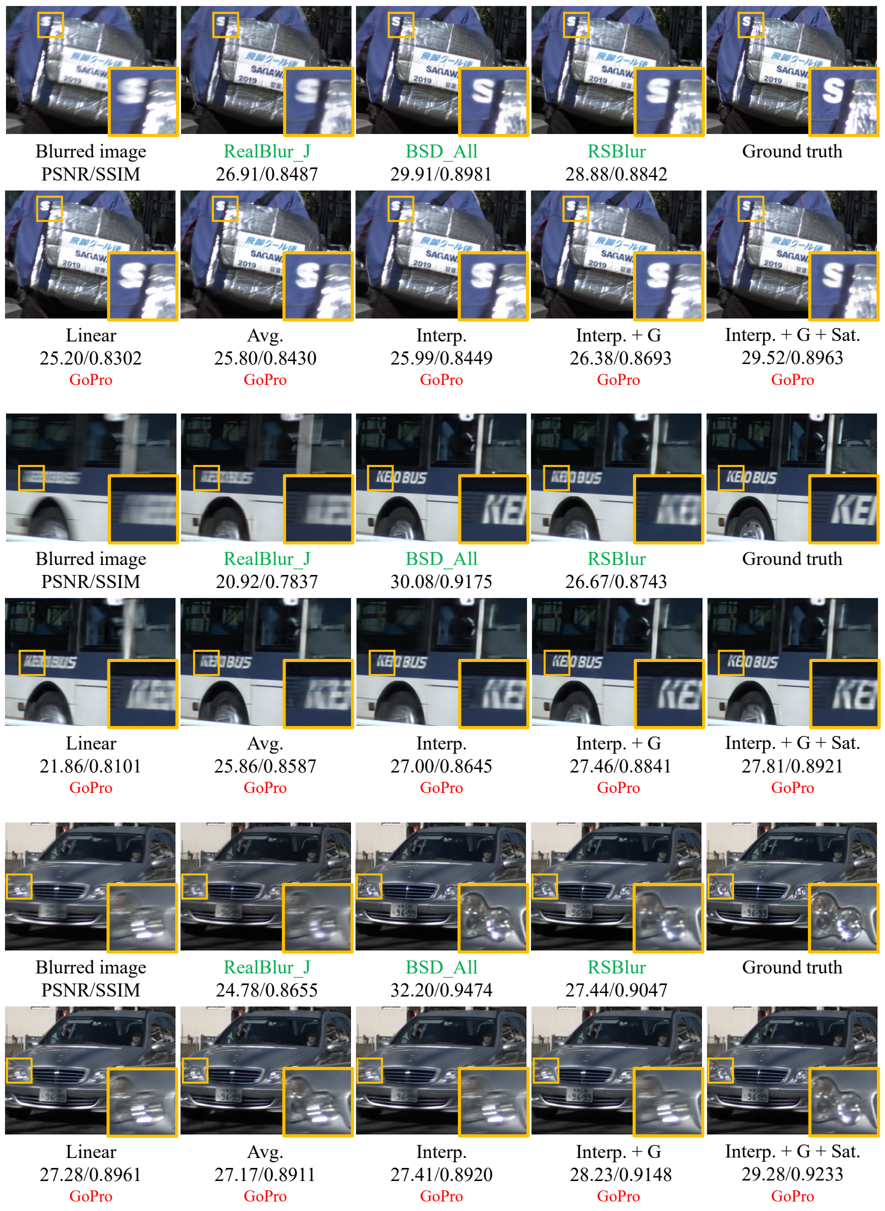


**Fig. 15.** Qualitative comparison on the BSD_All dataset [25, 26]. Green: Trained on real blurred images. Red: Trained on synthetic blurred images. Linear: Naïve averaging-based blur synthesis with linear CRF. Avg.: Naïve averaging-based blur synthesis. Interp.: Averaging-based blur synthesis using frame interpolation. G: Gaussian noise. Sat: Our saturation synthesis. All of synthesis methods except Linear consider gamma decoding and encoding.

## References

1. Abdelhamed, A., Brubaker, M.A., Brown, M.S.: Noise flow: Noise modeling with conditional normalizing flows. In: ICCV (October 2019) 13
2. Abdelhamed, A., Lin, S., Brown, M.S.: A high-quality denoising dataset for smartphone cameras. In: CVPR (June 2018) 9
3. Brooks, T., Mildenhall, B., Xue, T., Chen, J., Sharlet, D., Barron, J.T.: Unprocessing images for learned raw denoising. In: CVPR (June 2019) 6, 10
4. Chang, K.C., Wang, R., Lin, H.J., Liu, Y.L., Chen, C.P., Chang, Y.L., Chen, H.T.: Learning camera-aware noise models. In: ECCV. pp. 343–358 (2020) 13
5. Chen, G., Zhu, F., Ann Heng, P.: An efficient statistical method for image noise level estimation. In: ICCV (December 2015) 1
6. Chen, L., Zhang, J., Pan, J., Lin, S., Fang, F., Ren, J.S.: Learning a non-blind deblurring network for night blurry images. In: CVPR (June 2021) 12
7. Cho, S.J., Ji, S.W., Hong, J.P., Jung, S.W., Ko, S.J.: Rethinking coarse-to-fine approach in single image deblurring. In: ICCV. pp. 4641–4650 (October 2021) 2, 8, 9
8. Evangelidis, G.D., Psarakis, E.Z.: Parametric image alignment using enhanced correlation coefficient maximization. IEEE TPAMI **30**(10), 1858–1865 (Oct 2008) 6
9. Guo, S., Yan, Z., Zhang, K., Zuo, W., Zhang, L.: Toward convolutional blind denoising of real photographs. In: CVPR (June 2019) 8
10. Hu, Z., Cho, S., Wang, J., Yang, M.H.: Deblurring low-light images with light streaks. In: CVPR. pp. 3382–3389 (2014) 12
11. Jang, G., Lee, W., Son, S., Lee, K.M.: C2n: Practical generative noise modeling for real-world denoising. In: ICCV. pp. 2350–2359 (October 2021) 13
12. Köhler, R., Hirsch, M., Mohler, B., Schölkopf, B., Harmeling, S.: Recording and playback of camera shake: benchmarking blind deconvolution with a real-world database. In: ECCV. pp. 27–40 (2012) 14
13. Lai, W.S., Huang, J.B., Hu, Z., Ahuja, N., Yang, M.H.: A comparative study for single image blind deblurring. In: CVPR (June 2016) 14, 16, 17
14. Malvar, H., wei He, L., Cutler, R.: High-quality linear interpolation for demosaicing of bayer-patterned color images. In: 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing. vol. 3, pp. iii–485 (2004) 2, 6
15. Mitsunaga, T., Nayar, S.: Radiometric self calibration. In: CVPR. vol. 1, pp. 374–380 Vol. 1 (1999) 7
16. Pan, J., Hu, Z., Su, Z., Yang, M.H.: $l_0$ -regularized intensity and gradient prior for deblurring text images and beyond. IEEE TPAMI (2017) 12
17. Park, J., Lee, C., Kim, C.S.: Asymmetric bilateral motion estimation for video frame interpolation. In: ICCV. pp. 14539–14548 (October 2021) 6
18. Ren, W., Zhang, J., Ma, L., Pan, J., Cao, X., Zuo, W., Liu, W., Yang, M.H.: Deep non-blind deconvolution via generalized low-rank approximation. In: NeurIPS (2018) 12
19. Rim, J., Lee, H., Won, J., Cho, S.: Real-world blur dataset for learning and benchmarking deblurring algorithms. In: ECCV. pp. 184–201 (2020) 1, 6, 9, 10, 11, 14, 15, 19
20. Rowlands, D.A.: Color conversion matrices in digital cameras: a tutorial. Optical Engineering **59**(11), 1 – 36 (2020) 6
21. Tao, X., Gao, H., Shen, X., Wang, J., Jia, J.: Scale-recurrent network for deep image deblurring. In: CVPR (June 2018) 2, 6, 11, 14, 15

22. Wang, Z., Cun, X., Bao, J., Zhou, W., Liu, J., Li, H.: Uformer: A general u-shaped transformer for image restoration. In: CVPR (June 2022) 2
23. Zamir, S.W., Arora, A., Khan, S., Hayat, M., Khan, F.S., Yang, M.H.: Restormer: Efficient transformer for high-resolution image restoration. In: CVPR (June 2022) 2
24. Zamir, S.W., Arora, A., Khan, S., Hayat, M., Khan, F.S., Yang, M.H., Shao, L.: Multi-stage progressive image restoration. In: CVPR. pp. 14821–14831 (June 2021) 2
25. Zhong, Z., Gao, Y., Zheng, Y., Zheng, B.: Efficient spatio-temporal recurrent neural network for video deblurring. In: ECCV. pp. 191–207 (2020) 1, 11, 14, 15, 20
26. Zhong, Z., Gao, Y., Zheng, Y., Zheng, B., Sato, I.: Efficient spatio-temporal recurrent neural network for video deblurring. arXiv preprint arXiv:2106.16028 (2021) 1, 11, 14, 15, 20