Realistic Blur Synthesis for Learning Image Deblurring

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
http://cg.postech.ac.kr/research/RSBlur

Abstract. Training learning-based deblurring methods demands a tremendous amount of blurred and sharp image pairs. Unfortunately, existing synthetic datasets are not realistic enough, and deblurring models trained on them cannot handle real blurred images effectively. While real datasets have recently been proposed, they provide limited diversity of scenes and camera settings, and capturing real datasets for diverse settings is still challenging. To resolve this, this paper analyzes various factors that introduce differences between real and synthetic blurred images. To this end, we present RSBlur, a novel dataset with real blurred images and the corresponding sharp image sequences to enable a detailed analysis of the difference between real and synthetic blur. With the dataset, we reveal the effects of different factors in the blur generation process. Based on the analysis, we also present a novel blur synthesis pipeline to synthesize more realistic blur. We show that our synthesis pipeline can improve the deblurring performance on real blurred images.

Keywords: Realistic Blur Synthesis, Dataset and Analysis, Deblurring

1 Introduction

Motion blur is caused by camera shake or object motion during exposure, especially in a low-light environment that requires long exposure time. Image deblurring is the task of enhancing image quality by removing blur. For the past several years, numerous learning-based deblurring methods have been introduced and significantly improved the performance [22, 30, 15, 16, 36, 35, 8, 31, 34].

Training learning-based deblurring methods demands a significant amount of blurred and sharp image pairs. Since it is hard to obtain real-world blurred and sharp image pairs, a number of synthetically generated datasets have been proposed, whose blurred images are generated by blending sharp video frames captured by high-speed cameras [22, 21, 28, 43, 27, 11, 20]. Unfortunately, such synthetic images are not realistic enough, so deblurring methods trained on them often fail to deblur real blurred images [25].

To overcome such a limitation, Rim *et al.* [25] and Zhong *et al.* [40, 41] recently presented the RealBlur and BSD datasets, respectively. These datasets consist of real blurred and sharp ground truth images captured using specially

designed dual-camera systems. Nevertheless, coverage of such real datasets are still limited. Specifically, both RealBlur and BSD datasets are captured using *a single camera model*, Sony A7R3, and a machine vision camera, respectively [25, 40, 41]. As a result, deblurring models trained on each of them show significantly low performance on the other dataset, as shown in Sec. 6. Moreover, it is not easy to expand the coverage of real datasets as collecting such datasets require specially designed cameras and a tremendous amount of time.

In this paper, we explore ways to synthesize more realistic blurred images for training deblurring models so that we can improve deblurring quality on real blurred images without the burden of collecting a broad range of real datasets. To this end, we first present *RSBlur*, a novel dataset of real and synthetic blurred images. Then, using the dataset, we analyze the difference between the generation process of real and synthetic blurred images and present a realistic blur synthesis method based on the analysis.

Precise analysis of the difference between real and synthetic blurred images requires pairs of synthetic and real blurred images to facilitate isolating factors that cause the difference. However, there exist no datasets that provide both synthetic and real blurred images of the same scenes so far. Thus, to facilitate the analysis, the *RSBlur* dataset provides pairs of a real blurred image and a sequence of sharp images captured by a specially-designed high-speed dual-camera system. With the dataset, we can produce a synthetic blurred image by averaging a sequence of sharp images and compare it with its corresponding real blurred image. This allows us to analyze the difference between real and synthetic blurred images focusing on their generation processes. In particular, we investigate several factors that may degrade deblurring performance of synthetic datasets on real blurred images, such as noise, saturated pixels, and camera ISP. Based on the analysis, we present a method to synthesize more realistic blurred images. Our experiments show that our method can synthesize more realistic blurred images, and our synthesized training set can greatly improve the deblurring performance on real blurred images compared to existing synthetic datasets.

Our contributions are summarized as follows:

- We propose *RSBlur*, the first dataset that provides pairs of a real blurred image and a sequence of sharp images, which enables accurate analysis of the difference between real and synthetic blur.
- We provide a thorough analysis of the difference between the generation processes of real and synthetic blurred images.
- We present a novel synthesis method to synthesize realistic blurred images for learning image deblurring. While collecting large-scale real datasets for different cameras is challenging, our method offers a convenient alternative.

2 Related Work

Deblurring Methods Traditional deblurring methods rely on restrictive blur models, thus they often fail to deblur real-world images [26, 9, 33, 23, 29, 7,

18, 19]. To overcome such limitations, learning-based approaches that restore a sharp image from a blurred image by learning from a large dataset have recently been proposed [22, 30, 15, 16, 36, 35, 8, 31, 34]. However, they require a large amount of training data.

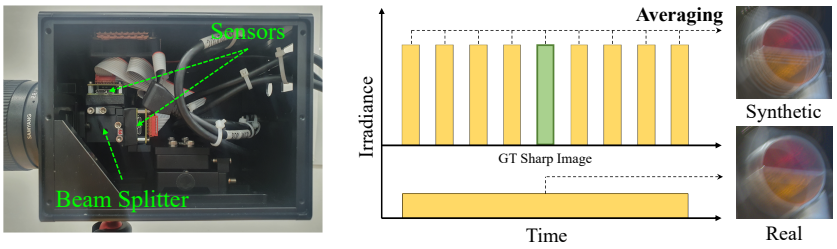
Deblurring Datasets For evaluation of deblurring methods, Levin *et al.* [18] and Köhler *et al.* [14] collected real blurred images by capturing images on the wall while shaking the cameras. Sun *et al.* [29] generate 640 synthetic blurred images by convolving 80 sharp images with eight blur kernels. Lai *et al.* [17] generate spatially varying blurred images from 6D camera trajectories and construct a dataset including 100 real blurred images. However, due to the small number of images, these datasets cannot be used for learning-based methods.

Several synthetic datasets using high-speed videos have been proposed for training learning-based methods. They capture high-speed videos and generate synthetic blurred images by averaging sharp frames. GoPro [22] is the most widely used dataset for learning-based deblurring methods. REDS [21] and DVD [28] provide synthetically blurred videos for learning video deblurring. Stereo Blur [43] consists of stereo blurred videos generated by averaging high-speed stereo video frames. HIDE [27] provides synthetic blurred images with bounding box labels of humans. To expand deblurring into high-resolution images, 4KRD [11] is presented, which consists of synthetically blurred UHD video frames. All the datasets discussed above, except for GoPro, use frame interpolation before averaging sharp images to synthesize more realistic blur [21, 28, 43, 27, 11]. HFR-DVD [20] uses high-speed video frames captured at 1000 FPS to synthesize blur without frame interpolation. However, all the aforementioned datasets are not realistic enough, thus deblurring networks trained with them often fail to deblur real-world blurred images.

Recently, real-world blur datasets [25, 40, 41] have been proposed. They simultaneously capture a real blurred image and its corresponding sharp image using a dual-camera system. Rim *et al.* [25] collected a real-world blur dataset in low-light environments. Zhong *et al.* [40, 41] proposed the BSD dataset containing pairs of real blurred and sharp videos. However, their performances degrade on other real images captured in different settings due to their limited coverage.

Synthesis of Realistic Degraded Images In the denoising field, synthesizing more realistic noise for learning real-world denoising has been actively studied [1, 6, 13, 32, 12, 4]. Abdelhamed *et al.* [1], Chang *et al.* [6], and Jang *et al.* [13] use generative models to learn a mapping from a latent distribution to a real noise distribution. Zhang *et al.* [39] and Wei *et al.* [32] propose realistic noise generation methods based on the physical properties of digital sensors. Guo *et al.* [12] and Brooks *et al.* [4] generate realistic noise by unprocessing arbitrary clean sRGB images, adding Poisson noise, and processing them back to produce noisy sRGB images. These methods show that more realistically synthesized noise datasets greatly improve the denoising performance of real-world noisy images.

Synthesis of Blurred Images A few methods have been proposed to synthesize blur without using high-speed videos [3, 38]. Brooks *et al.* [3] generate a blurred image from two sharp images using a line prediction layer, which esti-



(a) Our dual-camera system (b) Image acquisition process

Fig. 1. The dual-camera system and the acquisition process for simultaneously capturing a real blurred image and sharp images.

mates spatially-varying linear blur kernels. However, linear blur kernels cannot express a wide variety of real-world blur. Zhang *et al.* [38] use real-world blurred images without their ground-truth sharp images to train a GAN-based model to generate a blurred image from a single sharp image. However, their results are not realistic enough as their generative model cannot accurately reflect the physical properties of real-world blur.

3 RSBlur Dataset

Our proposed RSBlur dataset provides real blurred images of various outdoor scenes, each of which is paired with a sequence of nine sharp images to enable the analysis of the difference between real and synthetic blur. The dataset includes a total of 13,358 real blurred images of 697 scenes. For our analysis, we split the dataset into training, validation, and test sets with 8,878, 1,120, and 3,360 blurred images of 465, 58, and 174 scenes, respectively. Below, we explain the acquisition process and other details of the RSBlur dataset.

To collect the dataset, we built a dual camera system (Fig. 1(a)) as done in [25, 40, 41, 42]. The system consists of one lens, one beam splitter, and two camera modules with imaging sensors so that the camera modules can capture the same scene while sharing one lens. The shutters of the camera modules are carefully synchronized in order that the modules can capture images simultaneously (Fig. 1(b)). Specifically, one camera module captures a blurred image with a long exposure time. During the exposure time of a blurred image, the other module captures nine sharp images consecutively with a short exposure time. The shutter for the first sharp image opens when the shutter for the blurred image opens, and the shutter for the last sharp image closes when the shutter for the blurred image closes. The exposure time of the fifth sharp image matches with the center of the exposure time of the blurred image so that the fifth sharp image can be used as a ground-truth sharp image for the blurred one.

The blurred images are captured with a 5% neutral density filter installed in front of a camera module to secure a long exposure time as done in [40, 41]. The exposure times for the blurred and sharp images are 0.1 and 0.005 seconds, respectively. We capture images holding our system in hand so that blurred images can be produced by hand shakes. The captured images are geometrically

and photometrically aligned to remove misalignment between the camera modules as done in [25]. We capture all images in the camera RAW format, and convert them into the nonlinear sRGB space using a simple image signal processing (ISP) pipeline similar to [2] consisting of four steps: 1) white balance, 2) demosaicing, 3) color correction, and 4) conversion to the sRGB space using a gamma correction of sRGB space as a camera response function (CRF). More details on our dual-camera system and ISP are in the supplement.

4 Real *vs* Synthetic Blur

Using the RSBlur dataset, we analyze the difference between the generation process of real and synthetic blur. Specifically, we first compare the overall generation process of real and synthetic blur, and discover factors that can introduce the dominant difference between them. Then, we analyze each factor one by one and discuss how to address them by building our blur synthesis pipeline.

In the case of real blur, camera sensors accumulate incoming light during the exposure time to capture an image. During this process, blur and photon shot noise are introduced due to camera and object motion, and due to the fluctuation of photons, respectively. The limited dynamic range of sensors introduces saturated pixels. The captured light is converted to analog electrical signals and then to digital signals. During this conversion, additional noise such as dark current noise and quantization noise is added. The image is then processed by a camera ISP, which performs white balance, demosaicing, color space conversion, and other nonlinear operation that distort the blur pattern and noise distribution.

During this process, an image is converted through multiple color spaces. Before the camera ISP, an image is in the camera RAW space, which is device-dependent. The image is then converted to the linear sRGB space, and then to the nonlinear sRGB space. In the rest of the paper, we refer to the linear sRGB space as the linear space, and the nonlinear sRGB space as the sRGB space.

On the other hand, the blurred image generation processes of the widely used datasets, e.g., GoPro [22], DVD [28], and REDS [21], are much simpler. They use sharp images in the sRGB space consecutively captured by a high-speed camera. The sharp images are optionally interpolated to increase the frame rate [28, 21]. Then, they are converted to the linear space, and averaged together to produce a blurred image. The blurred image is converted to the sRGB space. For conversion between the linear to sRGB spaces, GoPro uses a gamma curve with $\gamma = 2.2$ while REDS uses a CRF estimated from a GOPRO6 camera.

Between the two processes described above, the main factors that cause the gap between synthetic and real blur include 1) discontinuous blur trajectories in synthetic blur, 2) saturated pixels, 3) noise, and 4) the camera ISP. In this paper, we analyze the effect of these factors one by one. Below, we discuss these factors in more detail.

Discontinuous Blur Trajectories The blur generation process of the GoPro dataset [22], which is the most popular dataset, captures sharp video frames at a high frame rate and averages them to synthesize blur. However, temporal gaps

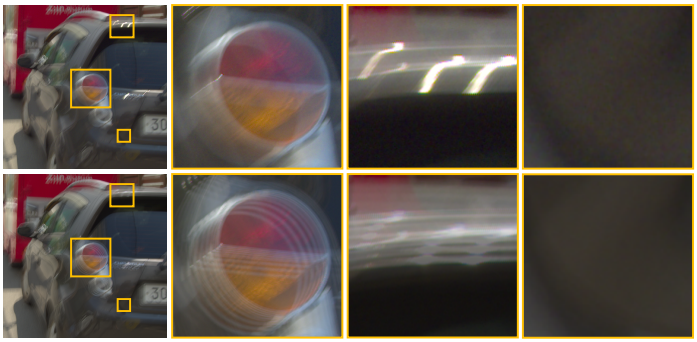


Fig. 2. The top row shows real blurred images and the bottom row shows the corresponding synthetic blurred images. Best viewed in zoom in.

between the exposure of consecutive frames cause unnatural discontinuous blur (Fig. 2(b)). While DVD [28] and REDS [21] use frame interpolation to fill such gaps, the effects of discontinuous blur and frame interpolation on the deblurring performance have not been analyzed yet.

Saturated Pixels While real-world blurred images may have saturated pixels (Fig. 2(c)) due to the limited dynamic range, previous synthetic datasets do not have such saturated pixels as they simply average sharp images. As saturated pixels in real blurred images form distinctive blur patterns from other pixels, it is essential to reflect them to achieve high-quality deblurring results [10].

Noise Noise is inevitable in real-world images including blurred images, especially captured by a low-end camera at night (Fig. 2(d)). Even for high-end sensors, noise cannot be avoided due to the statistical property of photons and the circuit readout process. In the denoising field, it has been proven important to model the realistic noise for high-quality denoising of real-world images [39, 32, 1, 6, 13, 4]. On the other hand, noise is ignored by the blur generation processes of the previous synthetic datasets [22, 21, 28, 43, 27, 11, 20], and its effect on deblurring has not been investigated. Our experiments in Sec. 6 show that accurate modeling of noise is essential even for the RealBlur dataset, which consists of images mostly captured from a high-end camera with the lowest ISO.

Camera ISP ISPs perform various operations, including white balancing, color correction, demosaicing, and nonlinear mapping using CRFs, which affect the noise distribution and introduce distortions [4, 5]. However, they are ignored by the previous synthetic datasets [22, 21, 28, 43, 27, 11, 20].

5 Realistic Blur Synthesis

To synthesize more realistic blur while addressing the factors discussed earlier, we propose a novel blur synthesis pipeline. The proposed pipeline will also serve as a basis for the experiments in Sec. 6 that study effect of each factor that degrades the quality of synthetic blur. Fig. 3 shows an overview of our blur

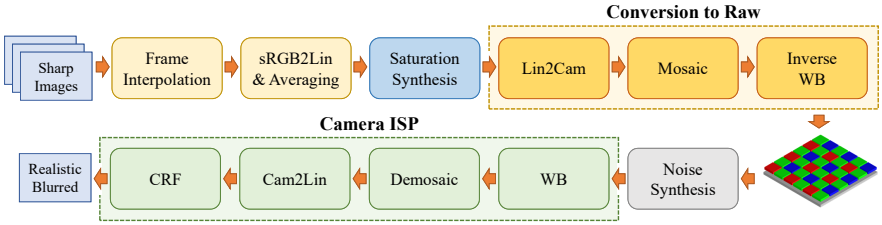


Fig. 3. Overview of our realistic blur synthesis pipeline. Lin2Cam: Inverse color correction, i.e., color space conversion from the linear space to the camera RAW space. WB: White balance. Cam2Lin: Color correction.

synthesis pipeline. Our pipeline takes sharp video frames captured by a high-speed camera as done in [22, 21, 28], and produces a synthetic blurred image. Both input and output of our pipeline are in the sRGB space. Below, we explain each step in more detail.

Frame Interpolation To resolve the discontinuity of blur trajectory, our pipeline adopts frame interpolation as done in [28, 21]. We increase nine sharp images to 65 images using ABME [24], a state-of-the-art frame interpolation method. In this step, we perform frame interpolation in the sRGB space to use an off-the-shelf frame interpolation method without modification or fine-tuning.

sRGB2Lin & Averaging To synthesize blur using the interpolated frames, we convert the images into the linear space, and average them to precisely mimic the real blur generation process. While the actual accumulation of incoming light happens in the camera RAW space, averaging in the camera RAW space and in the linear space are equivalent to each other as the two spaces can be converted using a linear transformation. Fig. 4(a) shows an example of the averaging of interpolated frames.

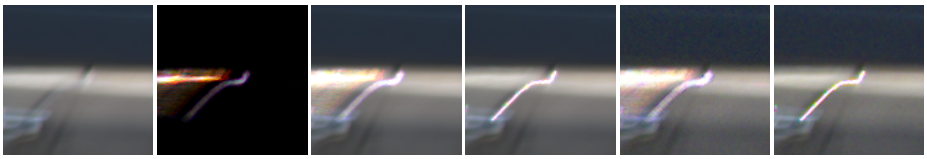
Saturation Synthesis In this step, we synthesize saturated pixels. To this end, we propose a simple approach. For a given synthetic blurred image B_{syn} from the previous step, our approach first calculates a mask M_i of the saturated pixels in the i-th sharp source image S_i of B_{syn} as follows:

$$M_i(x, y, c) = \begin{cases} 1, & \text{if } S_i(x, y, c) = 1 \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where (x, y) is a pixel position, and $c \in \{R, G, B\}$ is a channel index. S_i has a normalized intensity range $[0, 1]$. Then, we compute a mask M_{sat} of potential saturated pixels in B_{syn} by averaging M_i's. Fig. 4(b) shows an example of M_{sat}. Using M_{sat}, we generate a blurred image B_{sat} with saturated pixels as:

$$B_{sat} = \text{clip}(B_{syn} + \alpha M_{sat}) \tag{2}$$

where clip$(\cdot)$ is a clipping function that clips input values into $[0, 1]$, and α is a scaling factor randomly sampled from a uniform distribution $\mathcal{U}(0.25, 1.75)$.



(a) Averaging (b) M_{sat} (×3) (c) B_{sat} (d) B_{oracle} (e) Poisson (f) Real

Fig. 4. Generated images from our synthesis pipeline. (a) Averaging image of interpolated frames. (b) M_{sat} scaled by three times. (c)-(d) Examples of saturated images. (e)-(f) Synthetic noisy image and real image. The images except for (b) are converted into the sRGB space for visualization.

For the sake of analysis, we also generate blurred images with oracle saturated pixels. An oracle image B_{oracle} is generated as:

$$B_{oracle}(x,y,c) = \begin{cases} B_{real}(x,y,c), & \text{if } M_{sat}(x,y,c) > 0 \\ B_{syn}(x,y,c), & \text{otherwise.} \end{cases} \tag{3}$$

Our approach is simple and heuristic, and cannot reproduce the saturated pixels in real images due to missing information in sharp images. Specifically, while we resort to a randomly-sampled uniform scaling factor α, for accurate reconstruction of saturated pixels, we need pixel-wise scaling factors, which are impossible to estimate. Fig. 4(c) and (d) show examples of B_{sat} and B_{oracle} where the image in (c) looks different from the one in (d). Nevertheless, our experiments in Sec. 6 show that our approach still noticeably improves the deblurring performance on real blurred images.

Conversion to RAW In the next step, we convert the blurred image from the previous step, which is in the linear space, into the camera RAW space to reflect the distortion introduced by the camera ISP. In this step, we apply the inverse of each step of our ISP except for the CRF step in the reverse order. Specifically, we apply the inverse color correction transformation, mosaicing, and inverse white balance sequentially. As the color correction and white balance operations are invertible linear operations, they can be easily inverted. More details are provided in the supplement.

Noise Synthesis After the conversion to the camera RAW space, we add noise to the image. Motivated by [32, 39], we model noise in the camera RAW space as a mixture of Gaussian and Poisson noise as:

$$B_{noisy} = \beta_1(I + N_{shot}) + N_{read} \tag{4}$$

where B_{noisy} is a noisy image, and I is the number of incident photons. β_1 is the overall system gain determined by digital and analog gains. N_{shot} and N_{read} are photon shot and read noise, respectively. We model $(I + N_{shot})$ as a Poisson distribution, and N_{read} as a Gaussian distribution with standard deviation β_2. Mathematically, $(I + N_{shot})$ and N_{read} are modeled as:

$$(I + N_{shot}) \sim \mathcal{P}\left(\frac{B_{raw}}{\beta_1}\right)\beta_1, \quad \text{and} \tag{5}$$

$$N_{read} \sim \mathcal{N}(0, \beta_2) \tag{6}$$

where $\mathcal{P}$ and $\mathcal{N}$ denote Poisson and Gaussian distributions, respectively. B_{raw} is a blurred image in the camera RAW space from the previous step.

To reflect the noise distribution in the blurred images in the RSBlur dataset, we estimate the parameters β_1 and β_2 of our camera system as done in [39], where β_1 and β_2 are estimated using flat-field and dark-frame images, respectively. Refer to [39] for more details. The estimated values of β_1 and β_2 are 0.0001 and 0.0009, respectively. To cover a wider range of noise in our synthetic blurred images, we sample random parameter values β_1' and β_2' from $\mathcal{U}(0.5\beta_1, 1.5\beta_1)$ and $\mathcal{U}(0.5\beta_2, 1.5\beta_2)$, respectively. Then, using Eq. (5) and Eq. (6) with β_1' and β_2', we generate a noisy blurred image in the camera RAW space.

For the analysis in Sec. 6, we also consider Gaussian noise, which is the most widely used noise model. We obtain a noisy image with Gaussian noise as:

$$B_{noisy} = B + N_{gauss} \tag{7}$$

where B is an input blurred image and N_{gauss} is Gaussian noise sampled from $\mathcal{N}(0, \sigma)$, and σ is the standard deviation. As we include Gaussian noise in our analysis to represent the conventional noise synthesis, we skip the ISP-related steps (conversion to RAW, and applying camera ISP), but directly add noise to a blurred image in the sRGB space, i.e., we apply gamma correction to B_{sat} from the previous step, and add Gaussian noise to produce the final results. In our experiments, we randomly sample standard deviations of Gaussian noise from $\mathcal{U}(0.5\sigma', 1.5\sigma')$ where $\sigma' = 0.0112$ is estimated using a color chart image.

Applying Camera ISP Finally, after adding noise, we apply the camera ISP to the noisy image to obtain a blurred image in the sRGB space. We apply the same ISP described in Sec. 3, which consists of white balance, demosaicing, color correction, and CRF steps. Fig. 4(e) shows our synthesis result with Poisson noise and ISP distortions. As the example shows, our synthesis pipeline can synthesize a realistic-looking blurred image.

6 Experiments

In this section, we evaluate the performance of our blur synthesis pipeline, and the effect of its components on the RSBlur and other datasets. To this end, we synthesize blurred images using variants of our pipeline, and train a learning-based deblurring method using them. We then evaluate its performance on real blur datasets. In our analysis, we use SRN-DeblurNet [30] as it is a strong baseline [25], and requires a relatively short training time. We train the model for 262,000 iterations, which is half the iterations suggested in [30], with additional augmentations including random horizontal and vertical flip, and random rotation, which we found improve the performance. We also provide additional analysis results using another state-of-the-art deblurring method, MIMO-UNet [8], in the supplement.

6.1 Analysis using the RSBlur Dataset

We first evaluate the performance of the blur synthesis pipeline, and analyze the effect of our pipeline using the RSBlur dataset. Table 1 compares different variants of our blur synthesis pipeline. The method 1 uses real blurred images for training SRN-DeblurNet model [30], while the others use synthetic images for training. To study the effect of saturated pixels, we divide the RSBlur test set into two sets, one of which consists of images with saturated pixels, and the other does not, based on whether a blurred image has more than 1,000 non-zero pixels in M_{sat} computed from its corresponding sharp image sequence. The numbers of images in the sets with and without saturated pixels are 1,626 and 1,734, respectively. Below, we analyze the effects of different methods and components based on Table 1. As the table includes a large number of combinations of different components, we include the indices of methods that each analysis compares in the title of each paragraph.

Naïve Averaging (2 & 3) We first evaluate the performance of the naïve averaging approach, which is used in the GoPro dataset [22]. The GoPro dataset provides two sub-datasets: one of which applies gamma-decoding and encoding before and after averaging, and the other performs averaging without gamma-decoding and encoding. Thus, in this analysis, we also include two versions of naïve averaging. The method 2 in Table 1 is the most naïve approach, which uses naïve averaging and ignores CRFs. The method 3 also uses naïve averaging, but it uses a gamma correction of sRGB space as a CRF. The table shows that both methods perform significantly worse than the real dataset. This proves that there is a significant gap between real blur and synthetic blur generated by the naïve averaging approach of the previous synthetic dataset. The table also shows that considering CRF is important for the deblurring performance of real blurred images.

Frame Interpolation (3, 4, 5 & 6) We then study the effect of frame interpolation, which is used to fill the temporal gap between consecutive sharp frames by the REDS [21] and DVD [28] datasets. Methods 5 and 6 in Table 1 use frame interpolation. The method 6 adds synthetic Gaussian noise to its images as described in Sec. 5. Interestingly, the table shows that the method 5 performs worse than the method 3 without frame interpolation. This is because of the different amounts of noise in blurred images of methods 3 and 5. As frame interpolation increases the number of frames, more frames are averaged to produce a blurred image. Thus, a resulting blurred image has much less noise. The results of methods 6 and 4, both of which add Gaussian noise, verify this. The results show that frame interpolation performs better than naïve averaging when Gaussian noise is added.

Saturation (6, 7, 8, 9 & 10) To analyze the effect of saturated pixels, we first compare the method 6, which does not include saturated pixels whose values are clipped, and the method 7, which uses oracle saturated pixels. As shown by the results, including saturated pixels improves the deblurring quality by 0.12 dB. Especially, the improvement is large for the test images with saturated pixels (0.30 dB). Methods 8 and 10 use our saturation synthesis approach. The result

Table 1. Performance comparison among different blur synthesis methods on the RSBlur test set. 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	✓						32.53 / 0.8398	31.20 / 0.8313	33.78 / 0.8478
2		Linear					30.12 / 0.7727	28.67 / 0.7657	31.47 / 0.7793
3		sRGB					30.90 / 0.7805	29.60 / 0.7745	32.13 / 0.7861
4		sRGB			G		31.69 / 0.8258	30.18 / 0.8174	33.11 / 0.8336
5		sRGB	✓				30.20 / 0.7468	29.06 / 0.7423	31.27 / 0.7511
6		sRGB	✓		G		31.77 / 0.8275	30.28 / 0.8194	33.17 / 0.8352
7		sRGB	✓	Oracle	G		31.89 / 0.8267	30.58 / 0.8191	33.12 / 0.8338
8		sRGB	✓	Ours	G		31.83 / 0.8265	30.47 / 0.8187	33.12 / 0.8339
9		sRGB	✓	Oracle	G+P	✓	32.06 / 0.8315	30.79 / 0.8243	33.25 / 0.8384
10		sRGB	✓	Ours	G+P	✓	32.06 / 0.8322	30.74 / 0.8248	33.30 / 0.8391

(a) Blurred image PSNR/SSIM | (b) Method 1 29.81/0.8066 RSBlur | (c) Method 5 26.87/0.7129 | (d) Method 6 27.45/0.7938 | (e) Method 10 29.38/0.8027 Ours | (f) Ground truth

Fig. 5. Qualitative comparison of deblurring results on the RSBlur test set produced by models trained with different synthesis methods. (b)-(e) Methods 1, 5, 6 and 10 in Table 1. Best viewed in zoom in.

of the method 8 shows that, while it is worse than the method 7 (the oracle method), it still performs better the method 6, which does not perform saturation synthesis, especially for the test images with saturated pixels. Also, our final method (method 10) performs comparably to the oracle method (method 9). Both methods 9 and 10 achieve 32.06 dB for all the test images. This confirms the effectiveness of our saturation synthesis approach despite its simplicity.

Noise & ISP (5, 6, 7, 8, 9 & 10) We study the effect of noise and the ISP. To this end, we compare three different approaches: 1) ignoring noise, 2) adding Gaussian noise, and 3) adding Gaussian and Poisson noise with an ISP. The first approach corresponds to previous synthetic datasets that do not consider noise, such as GoPro [22], REDS [21] and DVD [28]. The second is the most widely used approach for generating synthetic noise in many image restoration tasks [37]. The third one reflects real noise and distortion caused by an ISP.

The table shows that, compared to the method 5 (No noise), the method 6 (Gaussian noise) performs significantly better by 1.57 dB. Moreover, a comparison between methods 7 and 8 (Gaussian noise) and methods 9 and 10 (Gaussian+Poisson noise with an ISP) shows that adding more realistic noise and distortion further improves the deblurring performance consistently.

Finally, our final method (method 10) achieves 32.06 dB, which is more than 1 dB higher than those of the naïve methods 2, 3, and 5. In terms of SSIM, our final method outperforms the naïve methods by more than 0.05. Compared to all the other methods, our final method achieves the smallest difference against

the method 1 which uses real blurred images. In terms of SSIM, our final method achieves 0.8332, which is only 0.0076 lower than that of the method 1. This proves the effectiveness of our method, and the importance of realistic blur synthesis.

Qualitative Examples Fig. 5(b)-(e) show qualitative deblurring results produced by models trained with different methods in Table 1. As Fig. 5(c) shows, a deblurring model trained with images synthesized using frame interpolation without noise synthesis fails to remove blur in the input blurred image. Adding Gaussian noise improves the quality (Fig. 5(d)), but blur still remains around the lights. Meanwhile, the method trained with our full pipeline (Fig. 5(e)) produces a comparable result to the method trained with real blurred images.

6.2 Application to Other Datasets

We evaluate the proposed pipeline on other datasets. Specifically, we apply several variants of our pipeline to the sharp source images of the GoPro dataset [22] to synthesize more realistic blurred images. Then, we train a deblurring model on synthesized images, and evaluate its performance on the RealBlur_J [25] and BSD [40, 41] datasets. The BSD dataset consists of three subsets with different shutter speeds. We use all of them as a single set, which we denote by BSD_All.

Limited Coverage of Real Datasets We examine the performance of real datasets on other real datasets to study the coverage of real datasets. To this end, we compare methods 1 and 2 in Table 2. The comparison shows that the performance of a deblurring model on one dataset significantly drops when trained on the other dataset. This proves the limitation of the existing real datasets and the need for a blur synthesis approach that can generate realistic datasets for different camera settings.

Improving GoPro The method 3 in Table 2 performs naïve averaging to the sharp source images in the GoPro dataset [22] without gamma correction. The method 4 performs gamma decoding, naïve averaging, and then gamma encoding. These two methods correspond to the original generation processes of GoPro. As the images in both RealBlur_J [25] and BSD [40, 41] datasets have blur distorted by the CRFs, the method 4 performs better. However, both of them perform much worse than the real-world blur training sets for both RealBlur_J and BSD_All.

The method 5 performs 0.01 dB worse than the method 4 on RealBlur_J. This again shows that frame interpolation without considering noise may degrade the deblurring performance as it reduces noise as discussed in Sec. 6.1. Adding Gaussian noise (method 6), and saturated pixels (method 7) further improves the deblurring performance on both test sets. For Gaussian noise, we simply add Gaussian noise with standard deviation $\sigma = 0.0112$.

The method 8 uses the noise and ISP parameters estimated for the RealBlur_J dataset [25]. The RealBlur_J dataset was captured using a Sony A7R3 camera, of which we can estimate the noise distribution, color correction matrix, and CRF. We use the method described in Sec. 5 for noise estimation. For the color correction matrix and CRF estimation, we refer the readers to our supplementary material. As the CRFs of the training set and RealBlur_J are different, the

Table 2. Performance comparison of different blur synthesis methods on the RealBlur_J [25] and BSD_All [40, 41] 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 parameters estimated from a Sony A7R3 camera, which was used for collecting the RealBlur dataset.

Blur Synthesis Methods							PSNR / SSIM	
No.	Training set	CRF	Interp.	Sat.	Noise	ISP	RealBlur_J	BSD_All
1	RealBlur_J						30.79 / 0.8985	29.67 / 0.8922
2	BSD_All						28.66 / 0.8589	33.35 / 0.9348
3	GoPro	Linear					28.79 / 0.8741	29.17 / 0.8824
4	GoPro	sRGB					28.93 / 0.8738	29.65 / 0.8862
5	GoPro	sRGB	✓				28.92 / 0.8711	30.09 / 0.8858
6	GoPro	sRGB	✓		G		29.17 / 0.8795	31.19 / 0.9147
7	GoPro	sRGB	✓	Ours	G		29.95 / 0.8865	31.41 / 0.9154
8	GoPro	sRGB, A7R3	✓	Ours	G+P	A7R3	30.32 / 0.8899	30.48 / 0.9060
9	GoPro_U	Linear					29.09 / 0.8810	29.22 / 0.8729
10	GoPro_U	sRGB					29.28 / 0.8766	29.72 / 0.8773
11	GoPro_U	sRGB			G		29.50 / 0.8865	30.22 / 0.8973
12	GoPro_U	sRGB		Ours	G		30.40 / 0.8970	30.31 / 0.8995
13	GoPro_U	sRGB, A7R3		Ours	G+P	A7R3	30.75 / 0.9019	29.72 / 0.8925

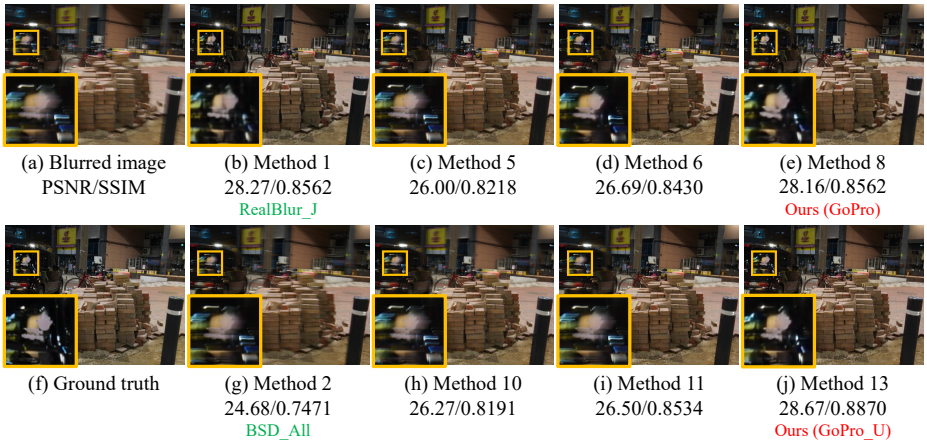


Fig. 6. Qualitative comparison of deblurring results on the RealBlur_J test set produced by models trained with different synthesis methods. (b)-(e) Methods 1, 5, 6 and 8 in Table 2. (g)-(j) Methods 2, 10, 11 and 13 in Table 2. Best viewed in zoom in.

method 8 uses different CRFs in different steps. Specifically, it uses gamma decoding with sharp source images of the GoPro dataset into the linear space, and in the last step of our pipeline, it applies the estimated CRF of Sony A7R3. The method 8 achieves 30.32 dB for the RealBlur_J dataset, which is much higher than 28.93 dB of the method 4. This proves that our blur synthesis pipeline reflecting the noise distribution and distortion caused by the ISP improves the quality of synthetic blur. It is also worth mentioning that the method 8 performs worse than the method 7 on the BSD_All dataset because the camera ISP of BSD_All is different from that of RealBlur_J. This also shows the importance of correct camera ISP parameters including CRFs, and explains why real datasets

perform poorly on other real datasets. Fig. 6(c)-(e) show results of deblurring methods trained on the GoPro dataset with different methods in Table 2.

Convolution-Based Blur Synthesis Our pipeline also applies to convolution-based blur synthesis and improves its performance as well. To verify this, we build a dataset with synthetic blur kernels as follows. For each sharp image in the GoPro dataset, we randomly generate ten synthetic blur kernels following [25] in order that we can convolve them with sharp images to synthesize blurred images instead of frame interpolation and averaging. We also compute saturation masks M_{sat} by convolving the masks of saturated pixels in each sharp image with the synthetic blur kernels. We denote this dataset as GoPro_U. Methods 9 to 13 in Table 2 show different variants of our pipeline using GoPro_U. For these methods, except for the frame interpolation and averaging, all the other steps in our pipeline are applied in the same manner.

In Table 2, methods 9 and 10 perform much worse than methods 1 and 2, which use real blurred images. Methods 11 and 12 show that considering Gaussian noise ($\sigma = 0.0112$) and saturated pixels significantly improves the performance. Finally, the method 13 that uses our full pipeline, achieves 30.75 dB in PSNR, which is only 0.04 dB lower than that of the method 1, and achieves 0.9019 in SSIM, which is 0.003 higher than that of the method 1. This shows that our pipeline is also effective for the convolution-based blur model. Fig. 6(h)-(j) show results of deblurring methods trained on the GoPro_U dataset with different methods in Table 2.

7 Conclusion

In this paper, we presented the RSBlur dataset, which is the first dataset that provides pairs of a real-blurred image and a sequence of sharp images. Our dataset enables accurate analysis of the difference between real and synthetic blur. We analyzed several factors that introduce the difference between them with the dataset and presented a novel blur synthesis pipeline, which is simple but effective. Using our pipeline, we quantitatively and qualitatively analyzed the effect of each factor that degrades the deblurring performance on real-world blurred images. We also showed that our blur synthesis pipeline could greatly improve the deblurring performance on real-world blurred images.

Limitations and Future Work Our method consists of simple and heuristic steps including a simple ISP and a mask-based saturation synthesis. While they improve the deblurring performance, further gains could be obtained by adopting sophisticated methods for each step. Also, there is still the performance gap between real blur datasets and our synthesized datasets, and some other factors may exist that cause the gap. Investigating that is an interesting future direction.

Acknowledgements This work was supported by Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT1801-05 and Institute of Information & communications Technology Planning & Evaluation (IITP) grants (2019-0-01906, Artificial Intelligence Graduate School Program (POSTECH)) funded by the Korea government (MSIT) and the National Research Foundation of Korea (NRF) grants (2020R1C1C1014863) funded by the Korea government (MSIT).

References

1. Abdelhamed, A., Brubaker, M.A., Brown, M.S.: Noise flow: Noise modeling with conditional normalizing flows. In: ICCV (October 2019) 3, 6
2. Abdelhamed, A., Lin, S., Brown, M.S.: A high-quality denoising dataset for smartphone cameras. In: CVPR (June 2018) 5
3. Brooks, T., Barron, J.T.: Learning to synthesize motion blur. In: CVPR (June 2019) 3
4. Brooks, T., Mildenhall, B., Xue, T., Chen, J., Sharlet, D., Barron, J.T.: Unprocessing images for learned raw denoising. In: CVPR (June 2019) 3, 6
5. Cao, Y., Wu, X., Qi, S., Liu, X., Wu, Z., Zuo, W.: Pseudo-isp: Learning pseudo in-camera signal processing pipeline from A color image denoiser. arXiv preprint arXiv:2103.10234 (2021) 6
6. 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) 3, 6
7. Cho, S., Lee, S.: Convergence analysis of map based blur kernel estimation. In: ICCV. pp. 4818–4826 (Oct 2017) 3
8. 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) 1, 3, 9
9. Cho, S., Lee, S.: Fast motion deblurring. ACM TOG **28**(5), 145:1–145:8 (Dec 2009) 3
10. Cho, S., Wang, J., Lee, S.: Handling outliers in non-blind image deconvolution. In: ICCV (2011) 6
11. Deng, S., Ren, W., Yan, Y., Wang, T., Song, F., Cao, X.: Multi-scale separable network for ultra-high-definition video deblurring. In: ICCV. pp. 14030–14039 (October 2021) 1, 3, 6
12. Guo, S., Yan, Z., Zhang, K., Zuo, W., Zhang, L.: Toward convolutional blind denoising of real photographs. In: CVPR (June 2019) 3
13. 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) 3, 6
14. 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) 3
15. Kupyn, O., Budzan, V., Mykhailych, M., Mishkin, D., Matas, J.: Deblurgan: Blind motion deblurring using conditional adversarial networks. In: CVPR. pp. 8183–8192 (June 2018) 1, 3
16. Kupyn, O., Martyniuk, T., Wu, J., Wang, Z.: Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In: ICCV (Oct 2019) 1, 3
17. 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) 3
18. Levin, A., Weiss, Y., Durand, F., Freeman, W.T.: Understanding and evaluating blind deconvolution algorithms. In: CVPR. pp. 1964–1971 (2009) 3
19. Levin, A., Weiss, Y., Durand, F., Freeman, W.T.: Efficient marginal likelihood optimization in blind deconvolution. In: CVPR. pp. 2657–2664 (2011) 3
20. Li, D., Xu, C., Zhang, K., Yu, X., Zhong, Y., Ren, W., Suominen, H., Li, H.: Arvo: Learning all-range volumetric correspondence for video deblurring. In: CVPR. pp. 7721–7731 (June 2021) 1, 3, 6
21. Nah, S., Baik, S., Hong, S., Moon, G., Son, S., Timofte, R., Mu Lee, K.: Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In: CVPRW (June 2019) 1, 3, 5, 6, 7, 10, 11

22. Nah, S., Kim, T.H., Lee, K.M.: Deep multi-scale convolutional neural network for dynamic scene deblurring. In: CVPR (July 2017) 1, 3, 5, 6, 7, 10, 11, 12
23. Pan, J., Sun, D., Pfister, H., Yang, M.H.: Blind image deblurring using dark channel prior. In: CVPR. pp. 1628–1636 (2016) 3
24. Park, J., Lee, C., Kim, C.S.: Asymmetric bilateral motion estimation for video frame interpolation. In: ICCV. pp. 14539–14548 (October 2021) 7
25. 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, 2, 3, 4, 5, 9, 12, 13, 14
26. Shan, Q., Jia, J., Agarwala, A.: High-quality motion deblurring from a single image. ACM TOG **27**(3), 73:1–73:10 (Aug 2008) 3
27. Shen, Z., Wang, W., Lu, X., Shen, J., Ling, H., Xu, T., Shao, L.: Human-aware motion deblurring. In: ICCV (October 2019) 1, 3, 6
28. Su, S., Delbracio, M., Wang, J., Sapiro, G., Heidrich, W., Wang, O.: Deep video deblurring for hand-held cameras. In: CVPR. pp. 237–246 (July 2017) 1, 3, 5, 6, 7, 10, 11
29. Sun, L., Cho, S., Wang, J., Hays, J.: Edge-based blur kernel estimation using patch priors. In: ICCP (2013) 3
30. Tao, X., Gao, H., Shen, X., Wang, J., Jia, J.: Scale-recurrent network for deep image deblurring. In: CVPR (June 2018) 1, 3, 9, 10
31. 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) 1, 3
32. Wei, K., Fu, Y., Yang, J., Huang, H.: A physics-based noise formation model for extreme low-light raw denoising. In: CVPR (June 2020) 3, 6, 8
33. Xu, L., Jia, J.: Two-phase kernel estimation for robust motion deblurring. In: ECCV (2010) 3
34. 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) 1, 3
35. 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) 1, 3
36. Zhang, H., Dai, Y., Li, H., Koniusz, P.: Deep stacked hierarchical multi-patch network for image deblurring. In: CVPR (June 2019) 1, 3
37. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE TIP **26**(7), 3142–3155 (2017) 11
38. Zhang, K., Luo, W., Zhong, Y., Ma, L., Stenger, B., Liu, W., Li, H.: Deblurring by realistic blurring. In: CVPR (June 2020) 3, 4
39. Zhang, Y., Qin, H., Wang, X., Li, H.: Rethinking noise synthesis and modeling in raw denoising. In: ICCV. pp. 4593–4601 (October 2021) 3, 6, 8, 9
40. Zhong, Z., Gao, Y., Zheng, Y., Zheng, B.: Efficient spatio-temporal recurrent neural network for video deblurring. In: ECCV. pp. 191–207 (2020) 1, 2, 3, 4, 12, 13
41. 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, 2, 3, 4, 12, 13
42. Zhong, Z., Zheng, Y., Sato, I.: Towards rolling shutter correction and deblurring in dynamic scenes. In: CVPR. pp. 9219–9228 (June 2021) 4
43. Zhou, S., Zhang, J., Zuo, W., Xie, H., Pan, J., Ren, J.S.: Davanet: Stereo deblurring with view aggregation. In: CVPR (June 2019) 1, 3, 6