# Gyro-based Neural Single Image Deblurring

Heemin Yang[1] Jaesung Rim[2] Seungyong Lee[1] Seung-Hwan Baek[1,2] Sunghyun Cho[1,2]

POSTECH CSE[1] & GSAI[2]

{heeminid, jsrim123, leesy, shwbaek, s.cho}@postech.ac.kr

## Abstract

*In this paper, we present GyroDeblurNet, a novel single-image deblurring method that utilizes a gyro sensor to resolve the ill-posedness of image deblurring. The gyro sensor provides valuable information about camera motion that can improve deblurring quality. However, exploiting real-world gyro data is challenging due to errors from various sources. To handle these errors, GyroDeblurNet is equipped with two novel neural network blocks: a gyro refinement block and a gyro deblurring block. The gyro refinement block refines the erroneous gyro data using the blur information from the input image. The gyro deblurring block removes blur from the input image using the refined gyro data and further compensates for gyro error by leveraging the blur information from the input image. For training a neural network with erroneous gyro data, we propose a training strategy based on the curriculum learning. We also introduce a novel gyro data embedding scheme to represent real-world intricate camera shakes. Finally, we present both synthetic and real-world datasets for training and evaluating gyro-based single image deblurring. Our experiments demonstrate that our approach achieves state-of-the-art deblurring quality by effectively utilizing erroneous gyro data.*

## 1. Introduction

Blur caused by camera shakes severely degrades image quality as well as the performance of various computer vision tasks. To overcome the image degradation caused by blur, single image deblurring has been extensively studied for decades [5, 11, 14, 16, 33, 49, 53]. Recently, various deep neural network (DNN)-based approaches have been proposed and have shown significant performance improvements over classical approaches. [3, 6, 21, 22, 31, 44, 52, 54]. Nonetheless, they still fail on images with large blur due to the severe ill-posedness of deblurring.

To address the ill-posedness of image deblurring, several attempts have been made to exploit the gyro sensors that most smartphones are now equipped with. Classical approaches based on blur models use gyro data to guide blur kernel estimation [10, 14, 16, 29, 41, 42]. However, their performance is limited by restrictive blur models that struggle to handle noise, nonlinear camera response functions, saturated pixels, moving objects, and other challenges effectively. With the advancement of deep learning, DNN-based approaches have also been proposed to exploit gyro data [15, 23, 30, 34, 46]. Thanks to the remarkable capabilities of DNNs and the rich information from gyro sensors, these methods demonstrate superior performance compared to the classical gyro-based approaches.

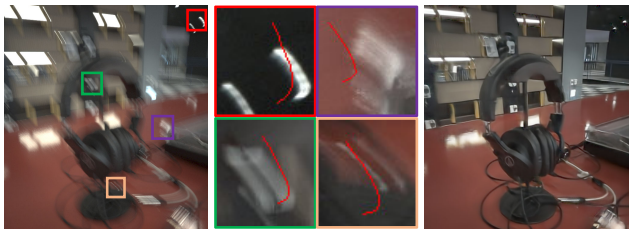


Figure 1. Our method shows robust performance even with errors in real-world gyro data. (a) A real-world blurry image. (b) Gyro data visualization onto the blurry image. The gyro data do not perfectly match the blur trajectories due to errors. (c) Our deblurred result.

However, recent DNN-based methods that utilize gyro data are still limited in handling real-world blurred images. First, gyro data from mobile devices such as smartphones usually contain a significant amount of noise caused by various sources [8, 16, 17]. Second, gyro sensors measure the angular velocity around their center. Therefore, recent gyro-based methods assume that the gyro sensor and the camera share the same center and that the camera is only affected by rotational motions. However, in practice, the positions of the camera and the gyro sensor differ, and the camera experiences more complex shakes. Third, real-world blurred images may have moving objects with different blur trajectories that cannot be captured by gyro sensors. All these factors cause the motion information encoded in the gyro data to be inconsistent with the blur in the blurred image (Fig. 1-(b)), making it challenging to exploit gyro data in

real-world image deblurring. We refer to such inconsistency between the gyro data and blur as *gyro error*.

In this paper, we propose *GyroDeblurNet*, a novel gyro-based single image deblurring approach that produces high-quality deblurring results even in the presence of gyro error (Fig. 1-(c)). To address gyro error, GyroDeblurNet adopts a carefully designed network architecture that includes novel gyro refinement blocks and gyro deblurring blocks. The gyro refinement block corrects gyro data errors using the blur information from the input image. On the other hand, the gyro deblurring block removes blur from the input image using the refined gyro data. Additionally, to further compensate for gyro error, the gyro deblurring block leverages both the blur information in the input image and the input gyro data. Despite these blocks, training a neural network with erroneous gyro data is challenging, as the network might be trained to ignore the input gyro data and rely solely on the input image. To resolve this issue, we propose a curriculum-learning-based training strategy [1].

To handle complex real-world camera shakes, we also present the *camera motion field*, a novel gyro data embedding scheme that can represent complex camera motions. Real-world blurry images often exhibit complex blur trajectories. Nonetheless, recent gyro-based approaches [15, 23, 30, 34] assume smooth camera motions and represent camera motions using one or two motion vectors per pixel or a couple of homographies for the entire image, which leads to low-quality deblurring results. To resolve this issue, our camera motion field is designed to represent complex camera motions using multiple vectors per pixel while maintaining a comparable memory footprint exploiting the spatially smooth nature of camera shakes.

Previous methods [15, 23, 30, 34] rely on synthetic datasets to train and evaluate the methods. However, their gyro data do not reflect real-world camera shakes that occur when capturing still-shot images either. Gyro data of previous methods were generated by random sampling [55] or obtained from the Visual-Inertial dataset [39]. However, the Visual-Inertial dataset was originally developed for visual odometry and SLAM, so its camera motions were collected from constantly moving cameras, which differ from camera shakes of still-shot images.

For training and evaluating gyro-based deblurring for real-world blurred images, we propose two datasets: *GyroBlur-Synth* and *GyroBlur-Real*. GyroBlur-Synth is a synthetic dataset for both training and evaluation. The dataset consists of sharp and blurred image pairs with their corresponding gyro data. The gyro data are acquired from a smartphone device to reflect real-world camera shakes. The blurred images are synthetically generated using the acquired gyro data and the blur synthesis process of RS-Blur [36] for realistic blurred images. GyroBlur-Real is a real-world blur dataset for qualitative evaluation with no ground-truth sharp images, and it provides real-world blurred images paired with gyro data collected from a smartphone.

We validate the performance of GyroDeblurNet on both synthetic and real-world datasets. Our experiments demonstrate that GyroDeblurNet clearly outperforms state-of-the-art deblurring methods both quantitatively and qualitatively. Our contributions can be summarized as follows:

- We propose a novel gyro-based single image deblurring approach, *GyroDeblurNet*. Extensive quantitative and qualitative evaluations show that GyroDeblurNet outperforms existing deblurring methods.
- To handle gyro error, we introduce a carefully designed network equipped with novel gyro refinement and gyro deblurring blocks. We also develop a novel gyro data embedding to represent complex camera shakes and propose a curriculum learning-based training scheme.
- We present synthetic and real image-gyro paired datasets with realistic camera motions, GyroBlur-Synth and GyroBlur-Real, for training and evaluating gyro-based single image deblurring methods.

## 2. Related Work

A variety of DNN-based single image deblurring approaches have recently been proposed [3, 6, 19, 21, 22, 31, 44, 45, 48, 52, 54]. To push the limit of single image deblurring, they introduce various network architectures and training schemes. Nevertheless, single image deblurring is still a challenging problem due to its high ill-posedness, and they still fail on images with large blur.

To resolve the ill-posedness of deblurring, there have been attempts to leverage inertial measurement sensors such as the gyro sensor and accelerometer as they provide valuable information on the camera motion. Joshi *et al*. [16] and Hu *et al*. [14] exploit the gyro sensor and accelerometer to estimate spatially-varying blur kernels. On the other hand, Park and Levoy [10] and Mustaniemi *et al*. [29] use only the gyro sensor to estimate blur kernels, because blur kernel estimation using accelerometer measurements requires estimating the depth of a scene, the device orientation to compensate the gravity effect, and the initial velocity of the device, which is often error-prone in practice. Moreover, rotational motion can be assumed as the dominant factor of camera shake blur as discussed by Whyte *et al*. [49].

In the deep learning era, there have been a few works that exploit gyro data. Mustaniemi *et al*. [30] propose a DNN-based approach that uses gyro data for the first time. They introduce a U-Net-based architecture [37] that takes a concatenation of a blurred image and the gyro data as input, which is also adopted by Lee *et al*. [23]. Ji *et al*. [15] adopt deformable convolutions to adjust convolution filters based on gyro data. Ren *et al*. [34] propose a transformer-based network that utilizes gyro data. However, these meth-

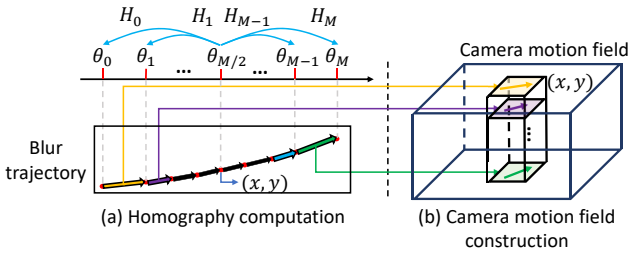


Figure 2. Camera motion field construction. (a) Computing homography and warped pixel coordinates. (b) Constructing camera motion field by stacking motion vectors.

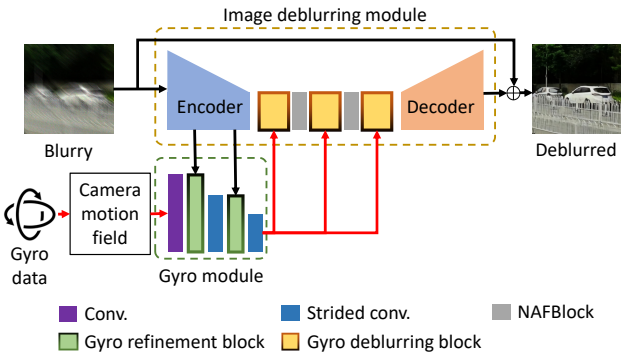


Figure 3. Network architecture of GyroDeblurNet.

ods assume accurate gyro data, neglecting potential errors. Moreover, they assume smooth camera motions and model camera motions using a vector field consisting of either one [30, 34] or two vectors [15] per pixel, or a couple of homographies [23] representing the beginning and end position of the camera during the exposure time. Unfortunately, such restrictive assumptions do not hold in real-world cases as discussed in Sec. 1.

Deblurring using gyro data can be framed as non-blind deconvolution, with gyro data serving as blur kernels. However, most non-blind deconvolution approaches assume accurate blur kernels without errors, making them unsuitable for erroneous gyro data [20, 24, 35, 50]. Some attempts have been made to address kernel errors in non-blind deconvolution. Vasu *et al*. [47] apply varying regularization strengths in a non-blind deblurring algorithm and use a neural network to combine results and refine artifacts induced by kernel errors. Nan *et al*. [32] tackle kernel error artifacts by estimating residual errors induced by kernel errors using a CNN. However, these approaches assume mild errors in kernels and fail to handle the significant errors presented in gyro data, as will be demonstrated in our experiments.

## 3. GyroDeblurNet

The goal of GyroDeblurNet is to estimate a sharp deblurred image $D$ from an input blurred image $B$ and its corresponding gyro data $G$ possibly with large errors. Specifically, we define $G$ as a sequence of gyro data such that $G = \{g_0, \cdots, g_{T-1}\}$, where $g_t$ is the $t$-th gyro data consisting of three angular velocities during the exposure time. $T$ is the number of gyro data, which is proportional to the exposure time. Instead of directly using $G$, we devise a novel gyro data embedding scheme named camera motion field to effectively handle complex camera shakes of an arbitrary exposure duration. In the rest of this section, we explain the gyro data embedding scheme, network architecture, and training strategy of GyroDeblurNet in detail.

### 3.1. Gyro Data Embedding

GyroDeblurNet first converts input gyro data sequence $G$ into a camera motion field $\mathcal{V}$ to handle temporally complex camera shakes of an arbitrary length $T$. The conversion process is illustrated in Fig. 2. A camera motion field has a fixed channel size $2M$ where $M$ is a hyperparameter representing the number of vectors in camera motion field, regardless of the length of the input gyro data to allow it to be fed to convolutional neural networks. To achieve this, we first resample the input gyro data sequence $G$ of an arbitrary length $T$ to $M+1$ samples using cubic-spline interpolation. We denote the resampled sequence as $G' = \{g'_0, \cdots, g'_M\}$ where $g'_m$ is a resampled gyro sample. The hyperparameter $M$ is set to an even number so that we have the same number of gyro data before and after the temporal center.

Then, assuming that the camera and the gyro shares the same center and there exists neither off-center rotational nor translational motions, we integrate the angular velocities and obtain $M+1$ camera orientations $\theta_m$, where $m = 0, \cdots, M$, including the camera orientations at the beginning and end of the exposure. Note that, despite our restrictive assumption on the camera motion, GyroDeblurNet can successfully perform deblurring thanks to its robustness to gyro errors. For each $\theta_m$, we compute a homography $H_m$ as $H_m = KR(\theta_m)K^{-1}$ where $K$ is the camera intrinsic matrix and $R(\theta_m)$ is the rotation matrix corresponding to $\theta_m$. To prevent unnecessary shift after deblurring, we chose the temporal center as a reference by assuming $\theta_{M/2}$ at the temporal center to be $(0, 0, 0)$ so that $H_{M/2}$ is an identity matrix when computing $\theta_m$.

Finally, from the obtained homographies, we compute a camera motion field $\mathcal{V}$, which illustrates how each pixel is blurred. Specifically, we define a camera motion field $\mathcal{V}$ as a tensor of size $W/s \times H/s \times 2M$ where $W$ and $H$ are the width and height of the input blurred image, and $s$ is a scaling factor. Then, at each position $(x, y)$ in $\mathcal{V}$, we compute the warped coordinates of $(x, y)$ by $H_m$ for all $m$ and compute the difference between temporally consecutive

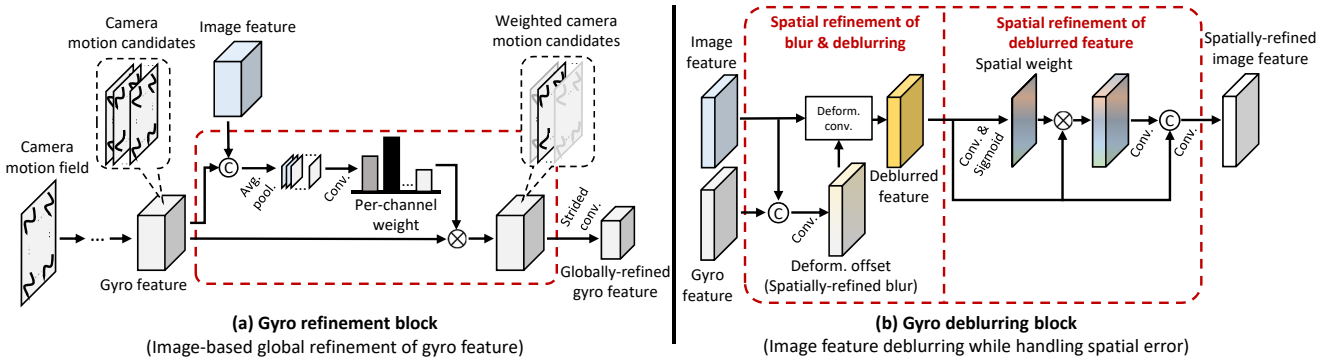


Figure 4. Detailed architectures of the modules in the GyroDeblurNet. (a) Architecture of the gyro refinement block. (b) Architecture of the gyro deblurring block.

coordinates to obtain $M$ vectors. By stacking the vectors for all the spatial positions, we construct $\mathcal{V}$. In our experiments, we set $M = 8$ and $s = 2$. Fig. 1-(b) visualizes camera motion field vectors onto a real-world blurry image.

Note that similar representations for gyro data based on vector fields have also been proposed by previous approaches [15, 30], as discussed in Sec. 2. However, unlike these methods that rely on only one or two vectors to represent camera shakes at each pixel, our camera motion field offers a more sophisticated representation. By adjusting the hyperparameter $M$, it can effectively capture temporally intricate camera shake patterns. Furthermore, our approach leverages the spatial smoothness of camera shakes, utilizing the hyperparameter $s$. As a result, it can provide a detailed representation of complex motions while maintaining a comparable memory footprint.

### 3.2. Network Architecture

GyroDeblurNet takes a blurred image $B$ and a camera motion field $\mathcal{V}$ as input and estimates a deblurred image $D$. The network consists of two modules: an image deblurring module and a gyro module (Fig. 3). The image deblurring module deblurs a blurred image $B$ with the aid of gyro data. It then produces a residual $R$ that is added back to $B$ to produce $D$. The image deblurring module adopts a U-Net architecture [37] and the NAFBlock [3] as its basic building block. The image deblurring module adopts gyro deblurring blocks in its bottleneck to perform deblurring using gyro features from the gyro network.

On the other hand, the gyro module takes a camera motion field $\mathcal{V}$ and refines it with the aid of image features from the image deblurring module. The gyro module consists of a convolution layer to embed an input camera motion field $\mathcal{V}$ into the feature space, gyro refinement blocks and strided convolution layers. The gyro refinement blocks refine gyro features with the aid of image features from the encoder of the image deblurring module. The gyro module progressively refines gyro features through gyro refinement blocks and strided convolution layers, and obtains a gyro feature of the same spatial size as the feature maps of the bottleneck in the image deblurring module. In the following, the gyro refinement block and the gyro deblurring block, which are the two key components of GyroDeblurNet, are explained in detail. For a detailed network architecture including feature dimensions, refer to the supplementary material.

**Gyro refinement block** Fig. 4-(a) illustrates the structure of the gyro refinement block. The camera motion information in the input camera motion field $\mathcal{V}$ may contain significant global errors caused by gyro sensor noise, unknown rotational centers, and missing translational motions, as visualized in Fig. 1-(b). The gyro refinement block is designed to compensate for these global errors using the blur information in the input image based on the following intuition.

The gyro refinement block takes a gyro feature computed from the input camera motion field $\mathcal{V}$ as input. This input gyro feature encodes multiple camera motion candidates across different channels, obtained by introducing perturbations to the input camera motion information in the previous layers. To globally refine the gyro feature, the gyro refinement block adaptively selects the channels of the gyro feature that are consistent with the blur information in the input image. Specifically, the gyro refinement block takes an image feature from the image deblurring module as an additional input and concatenates it with the input gyro feature. It then computes channel-wise weights [13] through global average pooling and $1 \times 1$ convolution layers, and multiplies them to the gyro feature. Following the gyro refinement block, a strided convolution layer further refines the gyro feature while reducing its spatial resolution.

**Gyro deblurring block** The gyro deblurring block, illustrated in Fig. 4-(b), performs deblurring of the image feature using the gyro feature from the gyro module. The input image $B$ may have spatially-variant blur caused by camera shakes and moving objects. To effectively remove such blur, the gyro deblurring block consists of two sub-blocks: the

first sub-block performs spatial refinement of the blur information and deblurring of the image feature using the refined blur information, while the second sub-block further refines the deblurred image feature.

Specifically, the first sub-block concatenates the image feature and gyro feature, computing spatially-refined blur information in the form of deformable convolution kernel offsets. It then performs deblurring of the image feature using these offsets. The second sub-block further refines the deblurred image feature using spatial attentions [51]. Spatial attention weights are computed from the deblurred image block through convolution and sigmoid layers, and multiplied with the deblurred image feature, which is further processed through convolution and concatenation layers.

### 3.3. Training Strategy

To address the challenge of training a deblurring network with erroneous gyro data, we propose a curriculum learning-based training strategy. This approach, inspired by its success in handling noisy data in prior work [1, 38, 40, 43], aims to progressively guide the network in leveraging gyro information effectively despite inaccuracies.

Our training strategy initially trains the network with error-free gyro data, and gradually increases error in the training data. To this end, we first assume that, for each blurred image $B$, its corresponding error-free gyro data $G_{GT}$ is given as well as its ground-truth sharp image $S_{GT}$. From $G_{GT}$, we compute a noise-free camera motion field $\mathcal{V}_{clean}$ and a noisy camera motion field $\mathcal{V}_{noisy}$. We generate $\mathcal{V}_{noisy}$ by randomly perturbing the rotational center and adding noise to the gyro data $G_{GT}$. We then compute a blended camera motion field $\mathcal{V}_\alpha = (1-\alpha)\cdot\mathcal{V}_{clean} + \alpha\cdot\mathcal{V}_{noisy}$ where $\alpha$ is a blending parameter. We gradually increase $\alpha$ from 0 to 1 during training. For more details including the noisy camera motion field generation and the scheduling of $\alpha$, refer to the supplementary material.

## 4. GyroBlur Dataset

### 4.1. GyroBlur-Synth

GyroBlur-Synth provides 14,600 and 640 synthetically blurred images of size $720 \times 1280$ for the training and test sets. The dataset also provides ground-truth sharp images and the corresponding gyro data. To faithfully mimic real-world blurred images, the dataset covers various degradation sources, such as moving objects, noise, and saturation.

To build a dataset with realistic camera motions, we implemented an Android application that records gyro data on a mobile phone. We used a Samsung Galaxy S22 smartphone for collecting gyro data with the sampling rate of 200 Hz. We collected two sequences of gyro data for generating training and test datasets with realistic hand-shake motions. The gyro data sequences are recorded for 195 sec. and 60 sec., and have 39,116 and 12,065 samples, respectively. We then used the gyro data to synthesize blurry images. To this end, we used sharp frames of the 4KRD dataset [7]. To generate a blurry image, we first randomly sampled a sequence of 10 consecutive gyro data corresponding to the exposure time of $1/20$ seconds. We then interpolated them by eight times to synthesize a continuous blur trajectory, and computed homographies from the interpolated gyro data. Finally, we warped the sharp image using the homographies, and blended them to obtain a blurry image.

To create synthetic blurred images with moving objects that have different blur trajectories, we used a simple method. We select a single instance from the COCO dataset [25] and assume it moves at a constant speed in a blurry image. We randomly choose its position and the direction and speed of motion. We then add the object to the warped sharp images before blending them during the blur synthesis process described earlier.

To reflect real-world noise and saturated pixels, we also adopt the blur synthesis pipeline of RSBlur [36]. Specifically, we synthesize blur in the linear sRGB space following the aforementioned process. Then, we synthesize saturated pixels, convert the image into the camera RAW space, synthesize noise, and convert the noisy RAW image to the linear sRGB image. We use the camera parameters of a Samsung Galaxy S22 ultra-wide camera, and the noise distribution estimated from it. We refer the readers to [36] for the detailed blur synthesis process.

To generate noisy camera motion fields, we add gyro errors with gyro data noise and randomly perturbed rotation centers. To obtain realistic gyro noise, we measured the gyro noise distribution from a stationary smartphone (Samsung Galaxy S22) placed on a table. Gyro noise is modeled as a Gaussian distribution for each rotation axis independently. For more details on the dataset, we refer the readers to the supplementary material.

Our synthetic dataset generation requires minimal device-specific input, brief gyro sequences, and simple camera calibration, significantly reducing manual data collection. We capture sufficient gyro data within minutes and obtain the necessary camera parameters for the RSBlur pipeline from a few calibration images. While we do not explicitly model the gyro's position relative to the camera, this is implicitly addressed as gyro errors such as rotational center error. Our experiments demonstrate effective blur removal regardless of these errors.

### 4.2. GyroBlur-Real

GyroBlur-Real consists of two subsets: GyroBlur-Real-S and GyroBlur-Real-H. GyroBlur-Real-S was collected using a Samsung Galaxy S22, which was also used for constructing the GyroBlur-Synth dataset. We use GyroBlur-Real-S as the primary dataset to evaluate the performance

| Method | Models | GyroBlur-Synth | | GyroBlur-Real-S | | Param. (M) | MACs (G) | Time (s) |
|---|---|---|---|---|---|---|---|---|
| | | PSNR ↑ | SSIM ↑ | NIQE ↓ | TOPIQ ↑ | | | |
| Single-image deblurring | MPRNet [52] | 25.03 | 0.7081 | 5.25 | 0.397 | 20.13 | 10927 | 0.998 |
| | NAFNet [4] | 25.06 | 0.7085 | 5.27 | 0.409 | 17.11 | 227 | 0.106 |
| | AdaRevD-B (NAFNet) [27] | 25.61 | 0.7281 | 5.18 | 0.425 | 43.81 | 2658 | 1.661 |
| | Uformer-B [48] | 25.72 | 0.7334 | 4.80 | 0.431 | 50.88 | 2143 | 1.061 |
| | Stripformer [45] | 25.93 | 0.7398 | 4.71 | 0.456 | 19.71 | 2397 | 1.367 |
| | FFTformer [19] | 26.01 | 0.7481 | 4.98 | 0.434 | 16.56 | 1894 | 1.879 |
| Non-blind deblurring | Vasu *et al*. [47] | 19.63 | 0.4526 | 6.13 | 0.317 | 10.41 | - | 160.176 |
| | Nan *et al*. [32] | 22.22 | 0.5313 | 5.81 | 0.348 | 26.16 | 41170 | 27.069 |
| Gyro-based deblurring | DeepGyro [30] | 23.78 | 0.6649 | 5.64 | 0.381 | 31.03 | 769 | 0.068 |
| | EggNet [15] | 25.49 | 0.7266 | 5.18 | 0.413 | 6.34 | 1102 | 0.071 |
| | INformer [34] | 25.11 | 0.7103 | 5.29 | 0.408 | 24.88 | 1315 | 0.464 |
| | Ours | 27.28 | 0.7803 | 4.47 | 0.548 | 16.31 | 262 | 0.130 |

Table 1. Quantitative comparison on the GyroBlur-Synth and GyroBlur-Real-S. For the GyroBlur-Real-S, we use non-reference metrics (NIQE [28] and TOPIQ [2] trained on KonIQ-10k dataset [12]) for evaluation. Inference times were measured using a $720 \times 1280$ image. For models utilizing gyro data, inference times include both gyro data embedding construction and the deblurring process.

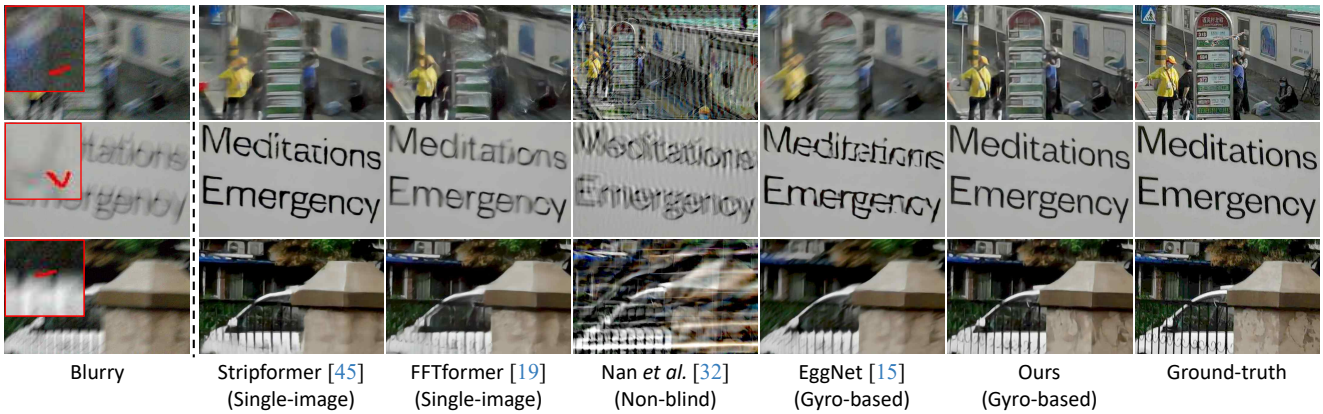


Figure 5. Qualitative comparison on GyroBlur-Synth. The red curves overlaid on the blurred images visualize the input gyro data.

of our approach on real-world images. On the other hand, GyroBlur-Real-H was collected using a Huawei P30 Pro, and is utilized for evaluating the generalization ability of our approach. GyroBlur-Real-S provides 100 real-world blurry images of size $4080 \times 3060$ along with their corresponding gyro data. GyroBlur-Real-H provides 17 real-world blurry images of size $5120 \times 3840$ and their gyro data. The images in both datasets are provided in DNG and JPEG.

## 5. Experiments

We implemented our network using PyTorch. We trained our network on randomly cropped $256 \times 256$ image patches from the GyroBlur-Synth dataset for 300 epochs with a batch size of 16. We used the Adam optimizer [18] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate was initially set to 0.0001 and reduced to 1e-7 using the cosine annealing scheduler [26]. We used the PSNR loss [3, 4] to train our model. We measured the computation times of all the models on a GeForce RTX 3090 GPU.

We evaluate the performance of our method using the test set of GyroBlur-Synth and GyroBlur-Real. Additional experiments and examples including visualization of the effect of gyro refinement and an analysis on the robustness to gyro errors are included in the supplementary material.

**Comparison with state-of-the-art methods** We compare GyroDeblurNet with state-of-the-art single image deblurring methods that use gyro data [15, 30, 34] and those that do not [4, 19, 27, 45, 48, 52]. We also compare GyroDeblurNet with non-blind deconvolution methods that are designed to handle kernel error [32, 47]. Since these methods assume uniform blur kernels, we extended them to handle spatially-varying blur kernels computed from gyro data by using patch-wise blur kernels [9]. For more details about our extensions of the non-blind deconvolution methods, we refer the readers to the supplementary material. We train all models with the training set of GyroBlur-Synth and evaluate the performance of all the models on the test set of GyroBlur-Synth and GyroBlur-Real-S.


Figure 6. Qualitative comparison on GyroBlur-Real-S. The red curves overlaid on the blurred images visualize the input gyro data.

| Model | GyroBlur-Synth PSNR ↑ / SSIM ↑ | GyroBlur-Real-S NIQE ↓ / TOPIQ ↑ |
|---|---|---|
| (a) Deblurring w/ no gyro data | 24.90 / 0.6997 | 5.23 / 0.391 |
| (b) Training w/ error-free gyro data | 24.94 / 0.7112 | 5.27 / 0.396 |
| (c) Deblurring w/ gyro data w/o any refine. | 25.47 / 0.7128 | 5.01 / 0.418 |
| (d) Gyro refine. w/o image features | 26.17 / 0.7465 | 4.75 / 0.447 |
| (e) Deform. conv. using only gyro features | 26.32 / 0.7542 | 4.76 / 0.460 |
| (f) GyroDeblurNet | 26.94 / 0.7667 | 4.61 / 0.511 |
| (g) GyroDeblurNet + curriculum learning | 27.28 / 0.7803 | 4.47 / 0.548 |

Table 2. Ablation study on the components of GyroDeblurNet. For the GyroBlur-Real-S, we use non-reference metrics (NIQE [28] and TOPIQ [2] trained on KonIQ-10K dataset [12]) for evaluation.

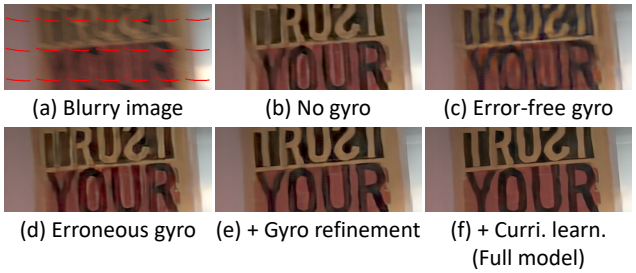


Figure 7. Qualitative results of the ablation study on a real-world blurry image.

The results are presented in Tab. 1. As shown in the table, GyroDeblurNet significantly outperforms all previous non-gyro-based methods. Specifically, compared to transformer-based approaches [19, 45, 48], our method requires substantially less computation and shorter computation time, while achieving more than 1 dB higher PSNR. Furthermore, our method surpasses all previous gyro-based methods by a large margin, despite requiring a comparable model size and much smaller computational overhead. The non-blind deblurring methods designed to handle blur kernel errors also perform significantly worse than our method due to large gyro errors. Nonetheless, our approach demonstrates superior performance even in the presence of large gyro errors.

Qualitative comparisons on GyroBlur-Synth and GyroBlur-Real-S are shown in Fig. 5 and in Fig. 6, respectively. As shown in the figures, the previous non-gyro-based methods fail to restore sharp details due to severe blur. Moreover, despite the input gyro data, the previous gyro-based methods also fail to produce sharp details as they are not robust to gyro errors. The non-blind deblurring methods that are designed to handle kernel errors show significant ringing artifacts due to large errors in the gyro data. On the other hand, our method successfully restores sharp images as our method can effectively exploit erroneous gyro data.

**Ablation study** We conducted an ablation study to investigate the effects of individual components in our method. To this end, we built several variants of GyroDeblurNet and compared their performances. Both quantitative and qualitative results are reported in Tab. 2 and Fig. 7, respectively.

Tab. 2-(a) corresponds to a model with no gyro modules, using only image features in the gyro deblurring blocks. Tab. 2-(b) represents our full model, but trained with error-free gyro data. Tab. 2-(c) is a model that is trained with erroneous gyro data but does not refine gyro features using image features. Specifically, in this model, gyro features are not concatenated with image features in both gyro refinement and gyro deblurring blocks. Tab. 2-(f) is our full model trained without curriculum learning, while Tab. 2-(g) is our full model trained with curriculum learning.

As shown in Tab. 2-(a), deblurring with no gyro data performs the worst, as it cannot benefit from the blur cue provided by gyro data. The model in (b) achieves only marginal improvement over (a), indicating the importance of reflecting gyro errors in training data. The results in (c), (d), (e) and (f) demonstrate the necessity of training with erroneous gyro data and gyro refinement using image features. Finally, curriculum learning in (g) further improves deblurring quality, as it helps the model to fully utilize erroneous gyro data.

Fig. 7 also highlights the impact of each component in our method. The results in Fig. 7-(b) and (c) reveal that training with error-free gyro data yields no performance gain over training without gyro data since error-free gyro data cannot reflect real-world gyro errors. In contrast, Fig. 7-(d) shows that training with erroneous gyro data aids in handling these errors but remains suboptimal. Gyro re-

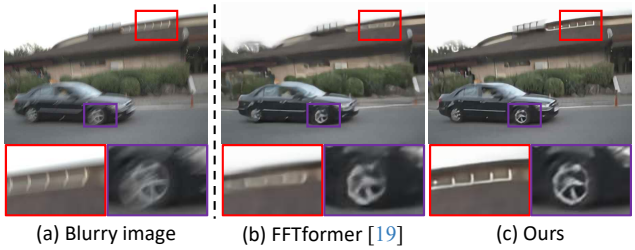


Figure 8. Results of a real-world blurred image with a moving object. (a) Blurry image and its background region and foreground object. (b) Result of FFTformer. (c) Result of GyroDeblurNet.

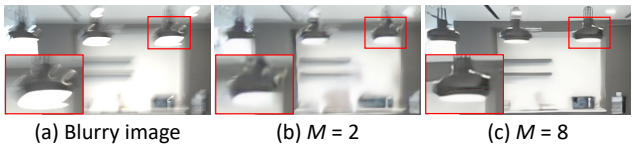


Figure 9. Effect of increasing $M$ for real-world complex motion blur. (a) Real-world blurry image with complex blur. (b) Result with $M = 2$. (c) Result with $M = 8$.

| $M$ | 2 | 4 | 8 | 16 |
|---|---|---|---|---|
| PSNR | 25.71 | 26.80 | 27.28 | 27.32 |
| SSIM | 0.7706 | 0.7651 | 0.7803 | 0.7811 |

Table 3. Effect of camera motion field hyperparameter $M$.

finement using image features in Fig. 7-(e) significantly improves image quality, while Fig. 7-(f) demonstrates how curriculum learning further enhances error handling.

**Moving objects** Another source of gyro error is moving objects with different blur trajectories. Fig. 8 shows a real-world example where the input blurry image has a moving object with a different blur trajectory. Due to severe blur, the result of FFTformer [19], the state-of-the-art non-gyro-based method, still has remaining blur in the background. In contrast, our method successfully restores sharp details on the background and shows comparable quality on the moving object thanks to its robustness to gyro errors.

**Camera motion field hyperparameter $M$** We investigate the impact of the hyperparameter $M$ that determines the temporal resolution of the camera motion field. As shown in Tab. 3, the deblurring quality improves gradually as $M$ increases. This highlights the importance of gyro data representation that can capture real-world intricate camera motions, and also explains the limited performance of previous gyro-based approaches [15, 30]. Fig. 9 also demonstrates the effect of increasing the number of vectors to handle complex real-world motion blur. However, increasing $M$ to 16 does not yield a significant performance gain while it doubles the memory consumption, indicating that $M = 8$ is sufficient to represent the camera shakes in most of the images in the test set of GyroBlur-Synth.


Figure 10. Blurry image taken with a Huawei P30 Pro device and its deblurred result. (a) A blurry image and its gyro errors. (b) Deblurred result of (a).

| Model | Stripformer [45] | FFTformer [19] | EggNet [15] | Ours |
|---|---|---|---|---|
| NIQE ↓ | 5.47 | 5.22 | 5.71 | 4.94 |
| TOPIQ ↑ | 0.451 | 0.459 | 0.432 | 0.521 |

Table 4. Quantitative evaluation on GyroBlur-Real-H using non-reference metrics [2, 28].

**Generalization to other devices** We test GyroDeblurNet's generalization ability by comparing it with top deblurring methods from Tab. 1 on GyroBlur-Real-H. All methods were trained on GyroBlur-Synth. The results in Tab. 4 show our approach outperforms the others, even on a different device. Fig. 10 also demonstrates that our approach successfully removes blur from an image captured by a different device, highlighting its generalization ability.

## 6. Conclusion

In this paper, we proposed GyroDeblurNet, a novel gyro-based single image deblurring method that can effectively restore sharp images with the help of gyro data. To fully exploit gyro data while considering real-world gyro error, we presented a novel gyro refinement block, a novel gyro deblurring block, and a curriculum learning-based training strategy. In addition, we also presented the camera motion field, a novel gyro embedding scheme to represent real-world camera motions. Finally, we proposed synthetic and real datasets where blurred images are paired with gyro data. Extensive quantitative and qualitative experiments demonstrate the effectiveness of our approach.

**Limitations and future work** Our method has a few limitations. While our method uses a fixed value for the hyperparameter $M$, the complexity of camera shakes may increase along with the exposure time, which means that a longer exposure time may require a larger value for $M$. Our method does not use the accelerometer, which is also equipped with most smartphones, and which may provide additional valuable information on the camera motion. Our image deblurring module adopts a relatively simple network architecture compared to recent non-gyro-based single image deblurring approaches, and this may limit the deblurring performance of our method. Extending our work to address these limitations would be an interesting future direction.

**Acknowledgements** This work was supported by the National Research Foundation of Korea (NRF) grants (RS-2023-NR077065, RS-2023-00280400, RS-2023-00211658, RS-2024-00438532) and by Institute of Information & communications Technology Planning & Evaluation (IITP) grant (RS-2019-II191906, Artificial Intelligence Graduate School Program (POSTECH)) funded by the Korea government (MSIT). This work was also supported by Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT1801-52.

## References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proc. ICML*, 2009. 2, 5

[2] Chaofeng Chen, Jiadi Mo, Jingwen Hou, Haoning Wu, Liang Liao, Wenxiu Sun, Qiong Yan, and Weisi Lin. Topiq: A top-down approach from semantics to distortions for image quality assessment. *IEEE TIP*, 2024. 6, 7, 8

[3] Liangyu Chen, Xin Lu, Jie Zhang, Xiaojie Chu, and Chengpeng Chen. Hinet: Half instance normalization network for image restoration. In *Proc. CVPR*, 2021. 1, 2, 4, 6

[4] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *Proc. ECCV*, 2022. 6

[5] Sunghyun Cho and Seungyong Lee. Fast motion deblurring. *ACM TOG*, 2009. 1

[6] Sung-Jin Cho, Seo-Won Ji, Jun-Pyo Hong, Seung-Won Jung, and Sung-Jea Ko. Rethinking coarse-to-fine approach in single image deblurring. In *Proc. ICCV*, 2021. 1, 2

[7] Senyou Deng, Wenqi Ren, Yanyang Yan, Tao Wang, Fenglong Song, and Xiaochun Cao. Multi-scale separable network for ultra-high-definition video deblurring. In *Proc. ICCV*, 2021. 5

[8] Dmytro Fedasyuk and Tetyana Marusenkova. Mems gyroscopes' noise simulation algorithm. In *Proc. Computer Science and Information Technologies*, 2020. 1

[9] Stefan Harmeling, Hirsch Michael, and Bernhard Schölkopf. Space-variant single-image blind deconvolution for removing camera shake. In *Proc. NeurIPS*, 2010. 6

[10] Sung Hee Park and Marc Levoy. Gyro-based multi-image deconvolution for removing handshake blur. In *Proc. CVPR*, 2014. 1, 2

[11] Michael Hirsch, Christian J Schuler, Stefan Harmeling, and Bernhard Schölkopf. Fast removal of non-uniform camera shake. In *Proc. ICCV*, 2011. 1

[12] Vlad Hosu, Hanhe Lin, Tamas Sziranyi, and Dietmar Saupe. Koniq-10k: An ecologically valid database for deep learning of blind image quality assessment. *IEEE TIP*, 2020. 6, 7

[13] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. CVPR*, 2018. 4

[14] Zhe Hu, Lu Yuan, Stephen Lin, and Ming-Hsuan Yang. Image deblurring using smartphone inertial sensors. In *Proc. CVPR*, 2016. 1, 2

[15] Seowon Ji, Jun-Pyo Hong, Jeongmin Lee, Seung-Jin Baek, and Sung-Jea Ko. Robust single image deblurring using gyroscope sensor. *IEEE Access*, 2021. 1, 2, 3, 4, 6, 8

[16] Neel Joshi, Sing Bing Kang, C Lawrence Zitnick, and Richard Szeliski. Image deblurring using inertial measurement sensors. *ACM TOG*, 2010. 1, 2

[17] Dennis Kim and Robert Thomas M'Closkey. Spectral analysis of vibratory gyro noise. *IEEE Sensors*, 2013. 1

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[19] Lingshun Kong, Jiangxin Dong, Jianjun Ge, Mingqiang Li, and Jinshan Pan. Efficient frequency domain-based transformers for high-quality image deblurring. In *Proc. CVPR*, 2023. 2, 6, 7, 8

[20] Dilip Krishnan and Rob Fergus. Fast image deconvolution using hyper-laplacian priors. In *Proc. NeurIPS*, 2009. 3

[21] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proc. CVPR*, 2018. 1, 2

[22] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *Proc. ICCV*, 2019. 1, 2

[23] Jeongmin Lee, Seo-Won Ji, Sung-Jin Cho, Jun-Pyo Hong, and Sung-Jea Ko. Deep learning-based deblur using gyroscope data. In *Proc. ICCE-Asia*, 2020. 1, 2, 3

[24] Anat Levin, Rob Fergus, Frédo Durand, and William T Freeman. Image and depth from a conventional camera with a coded aperture. *ACM TOG*, 2007. 3

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proc. ECCV*, 2014. 5

[26] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6

[27] Xintian Mao, Qingli Li, and Yan Wang. Adarevd: Adaptive patch exiting reversible decoder pushes the limit of image deblurring. In *Proc. CVPR*, 2024. 6

[28] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE SPL*, 2012. 6, 7, 8

[29] Janne Mustaniemi, Juho Kannala, Simo Särkkä, Jiri Matas, and Janne Heikkilä. Fast motion deblurring for feature detection and matching using inertial measurements. In *Proc. ICPR*, 2018. 1, 2

[30] Janne Mustaniemi, Juho Kannala, Simo Särkkä, Jiri Matas, and Janne Heikkila. Gyroscope-aided motion deblurring with deep networks. In *Proc. WACV*, 2019. 1, 2, 3, 4, 6, 8

[31] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proc. CVPR*, 2017. 1, 2

[32] Yuesong Nan and Hui Ji. Deep learning for handling kernel/model uncertainty in image deconvolution. In *Proc. CVPR*, 2020. 3, 6

[33] Jinshan Pan, Zhe Hu, Zhixun Su, and Ming-Hsuan Yang. Deblurring text images via l0-regularized intensity and gradient prior. In *Proc. CVPR*, 2014. 1

[34] Wenqi Ren, Linrui Wu, Yanyang Yan, Shengyao Xu, Feng Huang, and Xiaochun Cao. Informer: Inertial-based fusion transformer for camera shake deblurring. *IEEE TIP*, 2024. 1, 2, 3, 6

[35] William Hadley Richardson. Bayesian-based iterative method of image restoration. *Journal of the optical society of America*, 1972. 3

[36] Jaesung Rim, Geonung Kim, Jungeon Kim, Junyong Lee, Seungyong Lee, and Sunghyun Cho. Realistic blur synthesis for learning image deblurring. In *Proc. ECCV*, 2022. 2, 5

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, 2015. 2, 4

[38] Enver Sangineto, Moin Nabi, Dubravko Culibrk, and Nicu Sebe. Self paced deep learning for weakly supervised object detection. *IEEE TPAMI*, 2018. 5

[39] David Schubert, Thore Goll, Nikolaus Demmel, Vladyslav Usenko, Jörg Stückler, and Daniel Cremers. The tum vi benchmark for evaluating visual-inertial odometry. In *Proc. IROS*, 2018. 2

[40] Miaojing Shi and Vittorio Ferrari. Weakly supervised object localization using size estimates. In *Proc. ECCV*, 2016. 5

[41] Ondrej Sindelar and Filip Sroubek. Image deblurring in smartphone devices using built-in inertial measurement sensors. *Journal of Electronic Imaging*, 2013. 1

[42] Ondrej Sindelar, Filip Sroubek, and Peyman Milanfar. Space-variant image deblurring on smartphones using inertial sensors. In *Proc. CVPRW*, 2014. 1

[43] Samarth Sinha, Animesh Garg, and Hugo Larochelle. Curriculum by smoothing. In *Proc. NeurIPS*, 2020. 5

[44] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proc. CVPR*, 2018. 1, 2

[45] Fu-Jen Tsai, Yan-Tsung Peng, Yen-Yu Lin, Chung-Chi Tsai, and Chia-Wen Lin. Stripformer: Strip transformer for fast image deblurring. In *Proc. ECCV*, 2022. 2, 6, 7, 8

[46] Nisha Varghese, AN Rajagopalan, and Zahir Ahmed Ansari. Real-time large-motion deblurring for gimbal-based imaging systems. *IEEE JSTSP*, 2024. 1

[47] Subeesh Vasu, Venkatesh Reddy Maligireddy, and AN Rajagopalan. Non-blind deblurring: Handling kernel uncertainty with cnns. In *Proc. CVPR*, 2018. 3, 6

[48] Zhendong Wang, Xiaodong Cun, Jianmin Bao, Wengang Zhou, Jianzhuang Liu, and Houqiang Li. Uformer: A general u-shaped transformer for image restoration. In *Proc. CVPR*, 2022. 2, 6, 7

[49] Oliver Whyte, Josef Sivic, Andrew Zisserman, and Jean Ponce. Non-uniform deblurring for shaken images. In *Proc. CVPR*, 2010. 1, 2

[50] Norbert Wiener. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. The MIT press, 1949. 3

[51] Yong Xu, Ye Zhu, Yuhui Quan, and Hui Ji. Attentive deep network for blind motion deblurring on dynamic scenes. *Computer Vision and Image Understanding*, 2021. 5

[52] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *Proc. CVPR*, 2021. 1, 2, 6

[53] Haichao Zhang and David Wipf. Non-uniform camera shake removal using a spatially-adaptive sparse penalty. In *Proc. NeurIPS*, 2013. 1

[54] Hongguang Zhang, Yuchao Dai, Hongdong Li, and Piotr Koniusz. Deep stacked hierarchical multi-patch network for image deblurring. In *Proc. CVPR*, 2019. 1, 2

[55] Shuang Zhang, Ada Zhen, and Robert L Stevenson. A dataset for deep image deblurring aided by inertial sensor data. In *Proc. IS&T Symposium on Electronic Imaging*, 2020. 2