Ph.D. DISSERTATION

Deep Dynamic Scene Deblurring: New Datasets, Models, and Optimization

딥 러닝 기반 동적 영상 디블러링: 새로운 데이터셋, 모델, 그리고 최적화

BY

Seungjun Nah

AUGUST 2021

DEPARTMENT OF
ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Deep Dynamic Scene Deblurring: New Datasets, Models, and Optimization

딥 러닝 기반 동적 영상 디블러링: 새로운 데이터셋, 모델, 그리고 최적화

지도교수 이 경 무

이 논문을 공학박사 학위논문으로 제출함

2021년 8월

서울대학교 대학원

전기·정보공학부

나 승 준

나승준의 공학박사 학위 논문을 인준함

2021년 8월

위 원 장: _____

부위원장: _____

위 원: _____

위 원: _____

위 원: _____

Abstract

Obtaining a high-quality clean image is the ultimate goal of photography. In practice, daily photography is often taken in dynamic environments with moving objects as well as shaken cameras. The relative motion between the camera and the objects during the exposure causes motion blur in images and videos, degrading the visual quality. The degree of blur strength and the shape of motion trajectory varies by every image and every pixel in dynamic environments. The locally-varying property makes the removal of motion blur in images and videos severely ill-posed.

Rather than designing analytic solutions with physical modelings, using machine learning-based approaches can serve as a practical solution for such a highly ill-posed problem. Especially, deep-learning has been the recent standard in computer vision literature. This dissertation introduces deep learning-based solutions for image and video deblurring by tackling practical issues in various aspects.

First, a new way of constructing the datasets for dynamic scene deblurring task is proposed. It is nontrivial to simultaneously obtain a pair of the blurry and the sharp image that are temporally aligned. The lack of data prevents the supervised learning techniques to be developed as well as the evaluation of deblurring algorithms. By mimicking the camera image pipeline with high-speed videos, realistic blurry images could be synthesized. In contrast to the previous blur synthesis methods, the proposed approach can reflect the natural complex local blur from and multiple moving objects, varying depth, and occlusion at motion boundaries.

Second, based on the proposed datasets, a novel neural network architecture for single-image deblurring task is presented. Adopting the coarse-to-fine approach that is widely used in energy optimization-based methods for image deblurring, a multi-scale neural network architecture is derived. Compared with the single-scale model with similar complexity, the multi-scale model exhibits higher accuracy and faster speed.

Third, a light-weight recurrent neural network model architecture for video deblurring is proposed. In order to obtain a high-quality video from deblurring, it is important to exploit the intrinsic information in the target frame as well as the temporal relation between the neighboring frames. Taking benefits from both sides, the proposed intra-frame iterative scheme applied to the RNNs achieves accuracy improvements without increasing the number of model parameters.

Lastly, a novel loss function is proposed to better optimize the deblurring models. Estimating a dynamic blur for a clean and sharp image without given motion information is another ill-posed problem. While the goal of deblurring is to completely get rid of motion blur, conventional loss functions fail to train neural networks to fulfill the goal, leaving the trace of blur in the deblurred images. The proposed reblurring loss functions are designed to better eliminate the motion blur and to produce sharper images. Furthermore, the self-supervised learning process facilitates the adaptation of the deblurring model at test-time.

With the proposed datasets, model architectures, and the loss functions, the deep learning-based single-image and video deblurring methods are presented. Extensive experimental results demonstrate the state-of-the-art performance both quantitatively and qualitatively.

keywords: deblurring, dynamic scene, dataset, architecture, loss, deep learning

student number: 2014-21661

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recording visual snapshots of memorable moments is one of the long-lasting human desire. Before the cameras became popular, painting was the way to record the perception but it could take from minutes to several days to finish. With the advent of modern cameras, imaging sensors shortened the capturing time enabling hand-held cameras to capture our daily life. Still, if any scene changes happen during the exposure period, the taken photography gets blurred with textures being less recognizable as shown in Figure 1.1. Such motion blur is one of the most common artifacts in photography, degrading the visual quality significantly. In order to remove the motion blur and to recover the desired clean and sharp images, deblurring research has received much attention in computer vision community.

Motion blur occurs due to the shakes of camera and motions of the objects. When the blur is spatially uniform from translational camera shakes, a blurry image b can be considered a convolved output from the unknown sharp image s and the corresponding blur kernel k as

$$b = k * s + n, \tag{1.1}$$

where n is the pixel-wise image noise. Finding both the blur kernel and the latent sharp image simultaneously from the underdetermined system is already an ill-posed

Figure 1.1: **Examples of motion-blurred images**

problem. Casting the joint estimation of blur kernel and the sharp image as energy optimization framework, image priors were designed from natural image statistics [26, 93] to mitigate the ill-posedness and to reflect the preference on the desired solutions.

However, in general, camera is shaken in 3D space along with rotational movements, making the blur to be spatially non-uniform. Furthermore, freely moving objects complicates the blur trajectories and identifying the blur becomes more difficult. To express locally varying blur, (1.1) can be rewritten as

$$\mathbf{B} = \mathbf{K}\mathbf{S} + \mathbf{N}, \tag{1.2}$$

where $\mathbf{B}$, $\mathbf{S}$, $\mathbf{N}$ are the blurry and the sharp images and the noise in a vectorized form. $\mathbf{K}$ is a sparse matrix whose rows are the blur kernel for each pixel. The solution space for the spatially non-uniform blur is much larger than the spatially uniform case and the problem becomes severely ill-posed.

To cope with the camera shakes in 3D space for static scenes, [32, 33, 37, 111, 116] modeled the camera translation and rotation. Allowing more generic motion than camera shakes, [58, 40, 43] proposed dynamic motions by segmenting images from the detected motion. To overcome the dependency in segmentation quality in the previous works, [44] proposed a segmentation-free method by parameterizing the blur kernel as locally linear motion vectors.

Similarly to image deblurring research, early video deblurring methods assumed the scenes to be static [9, 60, 12, 57, 85], focusing on the camera motion. Extending the flexibility in motion modeling, [14, 5, 114] exploited the information from segmented scene layers. Motion modeling was generalized in [45] by using the bidirectional optical flow to estimate the pixel-wise blur kernels.

However, the previous approaches in image and video deblurring require the blur trajectories to be modeled from the physical formulations. The freely moving objects are hard to be modeled and parameterizing every feasible motion in a scene is extremely challenging. For such a highly ill-posed task, end-to-end learning methods could avoid the difficulty in physical modeling.

The main goal of this dissertation is to improve the dynamic scene deblurring by proposing deep learning-based approaches. To enable deep learning for deblurring problem, several steps are proposed, tackling different components. First, a novel way of constructing large-scale datasets for deblurring is proposed. Second, a convolution neural network architecture for image deblurring is presented by taking the virtue of the traditional optimization scheme in the deblurring task. Third, a recurrent neural network architecture for video deblurring is proposed by exploiting the intra-frame information as well as the inter-frame temporal relation. Lastly, a novel loss function is introduced to better optimize the deblurring network models by noticing the characteristics of clean images. The organization of the dissertation is described below with the contributions and the summarized details. The overview of the dissertation is summarized in Figure 1.2.

In Chapter 2, a way to generate deblurring datasets is introduced [72, 71]. A deblurring dataset requires a pair of blurry and the sharp image to capture the same moment with the difference in blur magnitude. In normal imaging conditions, recording exact the same scene with different conditions simultaneously is a nontrivial task. In contrast, mimicking camera image pipeline from a high-speed video can mitigate the problem. With consecutive frames being the slowly varying scenes, accumulat-

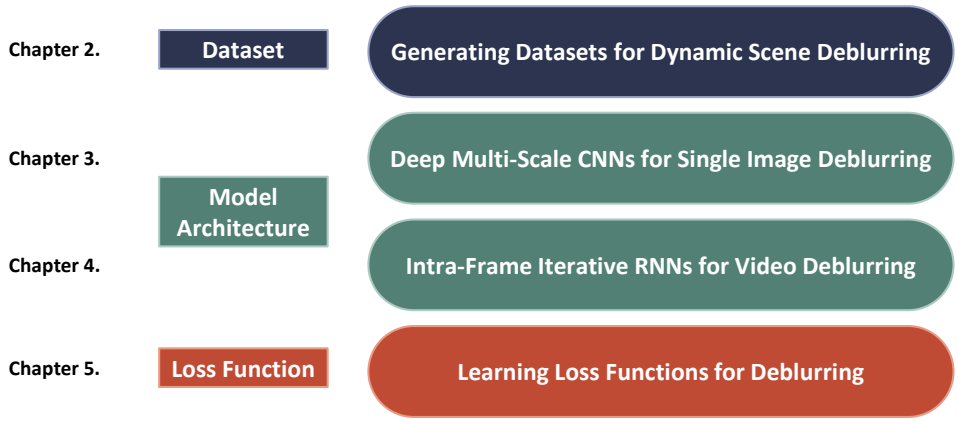


Figure 1.2: **Dissertation overview**

ing them over time simulates the realistic motion blur generation process. Picking the temporal center among the accumulated frames provides the corresponding sharp image for the synthesized blurry image. Thus, capturing high-frame-rate videos leads to deblurring dataset generation. Due to the lack of deblurring datasets, most previous deblurring results were only evaluated qualitatively. The proposed GOPRO and REDS datasets serve as test beds to benchmark deblurring algorithms [77, 76, 75] as well as the training datasets for deep learning models.

In Chapter 3, a convolutional neural network architecture is proposed for image deblurring problem [72]. As neural networks and deep learning became popular tools for computer vision tasks, convolutional networks are trained for deblurring task with the datasets proposed in chapter 2. However, even with the same number of weights, the achieved accuracy from training differs by the model architecture. To remove motion blur from an image with neural networks, the receptive field should be wide enough so that the scattered information along the blur trajectory could be handled. Simple stacking of sequential layers increase the receptive field proportional to the network depth. In order to increase the receptive field efficiently, traditional coarse-to-fine optimization scheme is adopted to build a multi-scale network architecture design. The multi-scale architecture first deblurs the given image in a coarse scale where the blur

magnitude is smaller. At the fine-scales, deblurring is performed by combining the input at the corresponding fine scale and the coarse deblurred image. They complement each other by providing the detailed texture and the reduced motion blur from the former process. The multi-scale networks not only perform better than the single-scale model with a similar complexity but also run in faster speed.

In Chapter 4, recurrent neural networks are adopted for video deblurring with additional operations [74]. Recurrent neural networks use the recurrence computation to let the hidden state propagate the information from the past to future frames. Due to the relevant information encoded in the hidden states, blur in the target frame can be better removed. While the hidden states act as a key for performance improvements in RNNs, they do not reflect the relation between the past and the current frames but only contain the information extracted from the past frames. To complement the hidden states with the relational information between the past and the current, the recurrence can be used to inject additional information. Given a fixed input blurry frame, the recurrence is reused to update the hidden states without moving forward in time. The hidden state is updated in the intra-frame iterations multiple times before the final deblurred frame is computed. Starting from a light-weight RNN cell, the proposed IFI-RNNs improve the deblurring performance in fast speed. In order to additionally improve the deblurring performance, the model is regularized at training time to favor more iterations only when significant improvement is available. Moreover, the model employing multiple cells exhibit further performance advances from the intra-frame iterations.

In Chapter 5, novel loss functions for deblurring are developed to predict sharper images [73]. In general, L1 or L2 distance functions are the most widely used loss functions to train neural networks in image restoration tasks. They optimize PSNR of the restored images but tend to produce rather blurry images as they prefer to find the average of possible solutions. Thus, applying them in deblurring task may yield imperfect removal of motion blur and partially leave the trace of blur. In order to better remove motion blur in the images, different types of loss functions should be designed

to train neural networks. Similarly to the ill-posedness in the deblurring task, synthesizing a realistic motion blur from only a single sharp image is another ill-posed problem as there could be infinitely many motion trajectories. In contrast, from an imperfectly deblurred image, it is possible to reconstruct the original blur by recognizing and amplifying the remaining blur with a learned reblurring neural network. By noticing the difference between the sharp and the incompletely deblurred images, a novel class of loss functions is designed in both supervised and self-supervised forms. The supervised reblurring loss compares the deblurred and the sharp images by trying to find the remaining blur footprint and magnifying it. The self-supervised reblurring loss inspects if an image is well deblurred by comparing it with its reblurred image. The deblurring models trained from the reblurring loss provide sharper images with improved perceptual quality. The self-supervised loss can further improve the perceptual quality from test-time adaptation.

The conclusion of the dissertation is provided in Chapter 6 with the consolidated contributions summary and the suggestions for future works.

Chapter 2

Generating Datasets for Dynamic Scene Deblurring

2.1 Introduction

Non-uniform blind motion deblurring for general dynamic scenes is a challenging computer vision problem. Motion blur occur not only from the camera shakes but also from the free motion of multiple objects. Conventional image and video deblurring methods tried to model the structure of blur by parameterizing the trajectory from physical formulations. For static scenes, camera movements with translations and rotations were considered to find the blur kernels [32, 37, 111]. Allowing more generic motions, [58] first proposed a method to handle locally varying blur by dividing the scene into multiple segments with motion variation. More smooth transition of blur kernels were proposed in [33, 40] but they were limited in handling abrupt changes of motion. In [43], dynamic scene deblurring was proposed so that the blur kernel, the latent image, and the motion segmentation was jointly estimated, considering a set of blur types. [44] generalized the approach and proposed segmentation-free method by modeling the blur as pixel-wise locally linear motion vectors.

Typically, energy minimization frameworks are designed with the data terms and the regularization terms to jointly find the latent sharp image and the spatially varying blur kernel. With $\mathbf{B}$ and $\mathbf{L}$ as the vectorized blurry and the latent sharp images and $\mathbf{K}$

as the blur kernel matrix, the energy is formulated as

$$E = E_{\text{data}}(\mathbf{B}, \mathbf{L}, \mathbf{K}) + E_{\text{reg}}(\mathbf{L}, \mathbf{K}). \tag{2.1}$$

The data term encodes the likelihood of $\mathbf{L}$ convolved with $\mathbf{K}$ reconstructing B by $\|\mathbf{B} - \mathbf{KL}\|$. To emphasize the object edges, $\|\nabla\mathbf{B} - \mathbf{K}\nabla\mathbf{L}\|$ is often used, too. As the solution space satisfying low E_{data} is very large, E_{reg} adopts prior knowledge on $\mathbf{L}$ and $\mathbf{K}$ for regularization. The energy terms are designed to follow the physical motion modeling in dynamic scenes with carefully designed priors [58, 33, 40, 43, 44]. The higher the degree of freedom in the motions in dynamic scenes, the more complex the energy formulations become. Generalizing the energy formulations for natural dynamics with high degree of freedom is prohibitively complicated and computationally expensive.

In contrast, letting a neural network model learn to deblur images can alleviate the complexity in system designs. With the advent of deep learning in computer vision [51, 101, 34, 35], neural networks were applied to many computer vision tasks including image restoration problems such as super-resolution [21] and denoising [125]. However, it has been difficult to apply such learning methods to image and video deblurring problems due to the lack of available datasets. Furthermore, quantitative benchmarks were limited as evaluation with sharp ground-truth was not available for real blurry images.

To facilitate early learning methods and benchmarks of deblurring algorithms, there were several attempts to synthesize motion blur. In [115] and [92], blur was synthesized by convolving an image with a low-pass filter but they were limited to spatially uniform cases. On the other hand, [50] used a robotic arm to replay camera motion on the same scenes to capture both the sharp and the blurry images. To prevent scene changes during the motion replay, printed photographs were attached on a planar board instead of capturing real scenes. [114] proposed layered motion models where the foreground and the background motions are different. Later, [54]

recorded 6D camera trajectories using inertial sensors of a cell phone and applied the non-uniform blur to internet-collected images. While these non-uniform blur synthesis methods [50, 114, 54] generate spatially non-uniform blur, they have several limitations. First, planar depths were assumed with single or double layers rather than reflecting the true depth of scenes. Second, most of the motions were limited to camera shakes without natural object motions. Third, they do not provide temporally center-aligned pairs of the blurry and the sharp images, making the reference-based evaluation with PSNR and SSIM to be less meaningful.

In this chapter, we propose to bring motion realism in dynamic scene blur synthesis by using high-speed videos. The blurring process can be modeled by the continuous integration of sharp images during the shutter exposure [105, 45, 47, 54]. We captured a sequence of sharp frames of a dynamic scene with high-speed cameras and averaged them to generate a blurry image. By mimicking the camera image pipeline, we propose GOPRO dataset [72] with videos captured in 240 fps To make the motion blur even more realistic in terms of temporal continuity, REDS dataset [71] is proposed with frame-rate upsampled to 1920 fps. REDS dataset further involves additional plausible artifacts combined with the motion blur to cast more challenging environments. Both the GOPRO and the REDS datasets provide temporally aligned pairs of the blurry and the sharp frames. The large-scale datasets enable not only the training of learning-based deblurring and also the quantitative comparison of different algorithms [77].

2.2 GOPRO dataset

Instead of trying to physically designing the blur kernel [54] to convolve on a sharp image, we choose to record the slowly varying sharp scenes to be integrated over time for blur image generation.

When taking a photograph, the camera sensor receives the light during the exposure and converts the collected signal into an RGB image. If the scene changes over

time, the sensor signal changes are accumulated to generate motion blur. The sensor stimulus at each moment can be considered a corresponding signal for the sharp image captured at an instant exposure. The integrated signal is then transformed into pixel values by nonlinear CRF (Camera Response Function). Thus, the blurring process can be approximated by accumulating frames from a high-frame-rate video as

$$B = g\left(\frac{1}{T}\int_{t=0}^{T}\hat{S}(t)dt\right) \simeq g\left(\frac{1}{M}\sum_{i=0}^{M-1}g^{-1}(S[i])\right), \tag{2.2}$$

by following the definition of definite integral. B is the blurry image, $S[i]$ is the i-th sharp frame in a video of length M and the recording duration T. The CRF g maps the sRGB frame $S[i]$ and the corresponding sensor signal $\hat{S}(t)$ at time t as $S[i] = g(\hat{S}(t))$. In practice, we only have the sRGB video frames while the original signal values and the CRF are unknown.

It is known that non-uniform deblurring becomes significantly difficult when nonlinear CRF is involved, and nonlinearity should be taken into account. However, currently, there are no CRF estimation techniques available for an image with spatially varying blur [102]. When the ground truth CRF is not given, a common practical method is to approximate CRF as a gamma curve with $\gamma = 2.2$ as follows, since it is known as an aproximated average of known CRFs [102].

$$g(x) = x^{1/\gamma}. \tag{2.3}$$

Thus, the sensor signal for the observed frame is obtained by $\hat{S}(T \times i/M) = g^{-1}(S[i])$ with the inverse CRF g^{-1} and we can synthesize the blurry image B by using (2.2).

In Figure 2.1, our kernel-free blurry image is compared with a conventional synthesized image with uniform blur kernel. Notably, the blur image generated by our method exhibits realistic and spatially varying blurs caused by the moving person and the static background while the conventional blur synthesized by convolution does not.

We used GOPRO4 Hero Black camera to generate GOPRO dataset. We took 240

(a) Sharp (b) Blur from convolution (c) Blur from accumulation

Figure 2.1: **Visual comparison of the blur by the synthesis method.** In this case, blur is mainly caused by the motion of person, leaving the background sharp. The blur kernel is non-uniform and complex. However, when the blurry image is synthesized by convolution with a uniform kernel, the background also gets blurred as if blur was caused by camera shake. To model dynamic scene blur, our kernel-free method is required. Our blur from accumulation accurately models the natural motion.

fps videos with the GOPRO camera and then averaged varying number (7 - 13) of successive sharp frames to produce blurs of different strengths. For example, averaging 15 frames simulates a photo taken at 1/16 shutter speed. Notably, the sharp latent image corresponding to each blurry one is defined as the mid-frame among the sharp frames that are used to make the blurry image. Finally, GOPRO dataset is composed of 3214 pairs of blurry and sharp images at 1280x720 resolution. 2103 image pairs are for training and the rest 1111 images are for testing. The proposed GOPRO dataset is publicly available online [1].

2.3 REDS dataset

We go a step further from the GOPRO dataset generation method to bring more realism in the blurry images and to improve the quality of the dataset. Our novel REDS dataset is proposed with REalistic and Dynamic Scenes of 720×1280 resolution high-quality video frames collected by ourselves. It has 30000 frames (300 sequences of length 100) with various contents, locations, natural and handmade objects. Moreover, we

[1] `https://seungjunnah.github.io/Datasets/gopro`

(a) REDS validation frames (b) REDS test frames

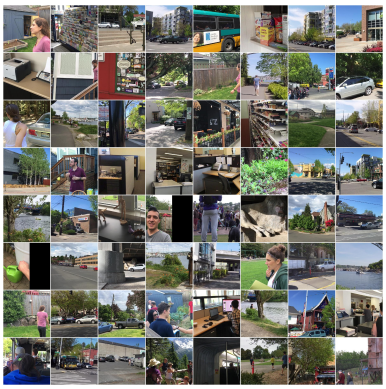
Figure 2.2: **Visualization of the proposed REDS validation and test set frames.** REDS contains 240, 30, 30 sequences for training, validation, test, respectively. Each sequence has 100 frame length.

organized the first example-based video deblurring and video super-resolution online challenges which used the REDS dataset. The dataset employs 4 types of degradations and corresponding competition tracks: motion blur, motion blur with compression artifacts, bicubic downscaling, and bicubic downscaling with motion blur. REDS dataset is for training and benchmarking example-based deblurring and super-resolution methods. REDS is intended to complement the existing video deblurring and SR datasets (see Fig. 2.3) to increase the content diversity and provide more realism in degradation, especially, motion blur.

The detailed data generation process is described below. The high-speed videos are recorded and the virtual frame rate in increased by interpolating the frames. Independently, the camera response function is measured to make blur synthesis process more realistic. After averaging the frames to simulate blur, additional artifacts are optionally added to generate more practical degradations.

Recording: We manually recorded 300 RGB video clips, paying attention to the quality of each frame, diversity of source contents (scenes and locations) and dynamics

(a) GOPRO dataset [72]



(b) DVD dataset [97]

(c) Real blurry videos [15]

(d) Vid4 dataset [63]

Figure 2.3: **Visualization of the other popular video datasets.** (a), (b), and (c) are for deblurring and (d) is for super-resolution.

of various motion. We used the GoPro HERO6 Black camera to record videos of 1080×1920 resolution at 120 fps. In contrast to the previous datasets for deblurring that captured videos in higher frame rate (240 fps) [72, 97], we choose slower frame rate for better image quality. Note that most consumer-level high-speed cameras don't access all of the sensor array cells during the readout time and performs interpolation to fill in the missing values. Under the limited computational power of the camera processors, decreasing the frame rate allows access to more sensor array elements, increasing the number of effective pixels per frame. Each frame remains sharp when the shutter speed is fast, however, the number of effective pixels is still less than

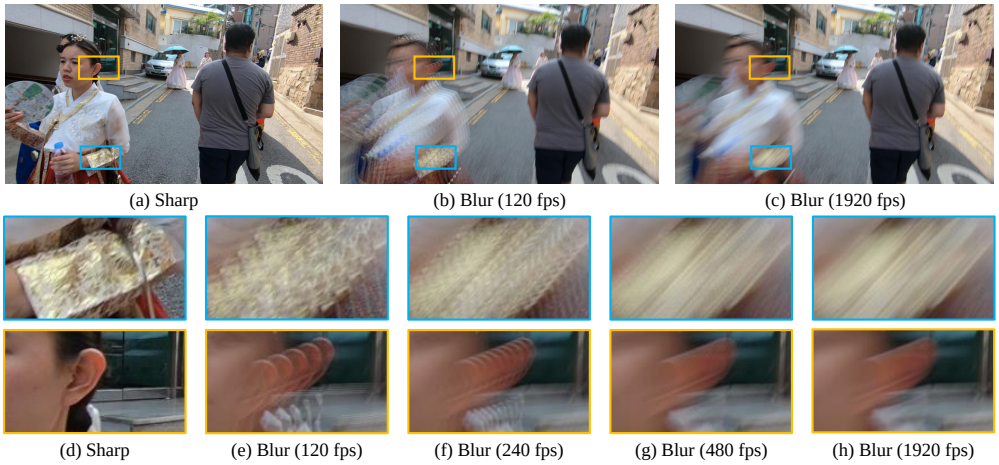


Figure 2.4: **Visual comparison of the synthesized blur by the virtual frame rate of video.** Averaging frames at 120 or 240 fps could cause unnatural blurs with ghost artifacts in case of large motion. The noted fps refers to the virtual frame rate of interpolated videos that are averaged to create blurry frames.

the full resolution. Also, noise or MPEG lossy compression could bring some visual artifacts.

Frame interpolation: Motion blur occurs due to the dynamics during the camera exposure and averaging the high-frame-rate video frames approximates the photograph taken at a longer exposure [72]. When the frame rate is not high enough, simply averaging frames may generate unnatural spikes or steps in the blur trajectory [112], especially when the spatial resolution is high and the motion is fast. To fill in the missing information between the frames, we employed a CNN trained to interpolate frames [80]. We chose a learned CNN instead of using optical flow to handle nonlinear motions and the warping artifacts. We increase the frame rate to virtual 1920 fps by recursively interpolating the frames. The effect of interpolation in the quality of blur is shown in Figure 2.4.

Calibration: When taking a picture, the sensor signal is converted to RGB pixels by a nonlinear CRF. Instead of using a theoretical average of all CRFs as in GOPRO dataset [72], to make REDS dataset, we calibrated the inverse CRF using [90] by using

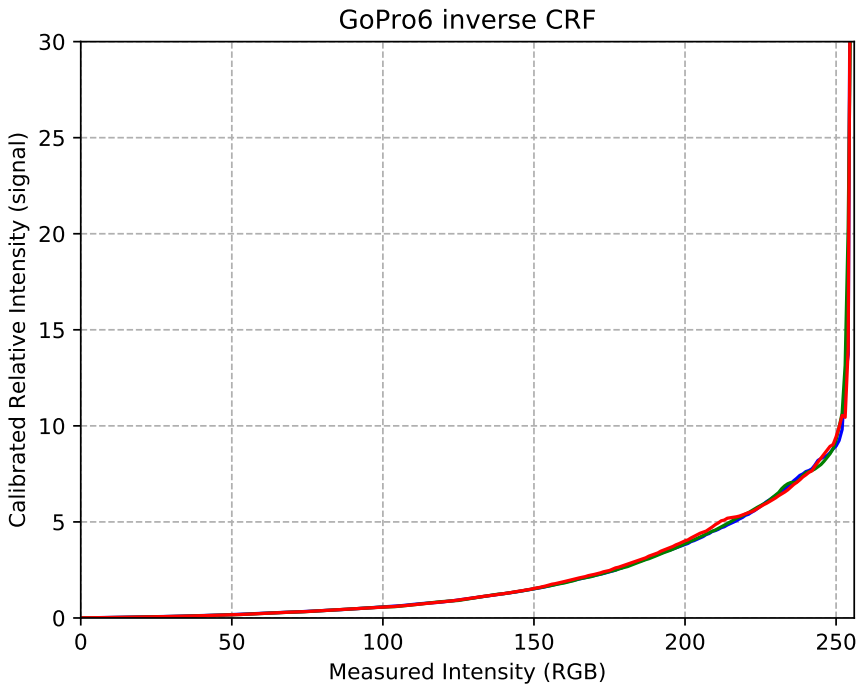


Figure 2.5: **Calibrated inverse camera response function of GoPro HERO6 Black for RGB channels.** It differs from the linear or gamma function assumptions from the previous datasets.

images captured at various exposures. As 8-bit sRGB representation saturates at value 255, the calibration could be inaccurate at higher pixel values ($p > 250$) when the calibration images are over-exposed. Hence, we replace the inverse CRF at $p > 250$ by appending a linear function having a slope of the inverse CRF at $p = 250$. Here, p denotes the RGB pixel value. We visualize the estimated CRF in Figure 2.5 and compare with linear [97] and gamma function [72] that are used for synthesizing blur.

Blur synthesis: We average the 1920 fps video frames to produce virtual 24 fps blurry video with duty cycle $\tau = 0.8$. The averaging is done in the signal space to mimic a camera imaging pipeline, using the estimated CRF and the inverse CRF. To further increase the per-pixel quality of the data, we suppress the noise and artifacts by down-

Figure 2.6: **Visualization of the REDS dataset and the provided degradations.** In NTIRE 2019 video deblurring challenge, motion blurs (Track 1) and compressed video (Track 2) data were provided. NTIRE 2019 video super-resolution challenge provided the low-resolution of the sharp (Track 1) and the blurry (Track 2) frames.

scaling both the synthesized blurry frames and the recorded sharp frames by $2/3$ to 720×1280 resolution. We used OpenCV function `resize` bicubic interpolation as it produces visually sharper results than MATLAB due to the different parameter values. There are 300 sequences in total, and each sequence contains 100 pairs of the blurry and sharp frame. We use those generated blurry videos as input for the NTIRE 2019 Video Deblurring Challenge Track 1: Clean.

Video compression: The above process was done to produce high-quality videos and blurs without realistic artifacts such as noise and compression. To promote the development of deblurring methods that apply to more realistic and common degradation, we compress the frames by saving the videos in mp4 (MPEG-4 Part 14) format. We used MATLAB `VideoWriter` to save the videos at 60% quality. Those compresed blurry videos are introduced to the NTIRE 2019 Video Deblurring Challenge: Track 2 Compression artifacts.

Downscaling: We also downscale the sharp and the blurry frames, respectively, to promote the development of example-based video super-resolution algorithms. They are employed in the NTIRE 2019 Video Super-Resolution Challenge: Track 1 Clean and Track 2 Blur. We used MATLAB function `imresize` bicubic interpolation at scale 4.

Diversity: We visited various countries, cities and towns, institutes and facilities, theme parks, festivals, palaces and castles, tourist attractions, historical places, zoos, stores, water parks, etc. to capture diverse scenes and objects. The contents include people from various nationalities, crowds, handmade objects, buildings, structures, artworks, furniture, vehicles, colorful textured clothes, and many other objects of different categories.

Partitions: After collecting and processing the REDS 300 video sequences, we computed the PSNR between the blurry and sharp frames. We split the REDS 300 sequences of frames into the train, validation, test sets. We randomly generated partitions of 240 train, 30 validation, and 30 test sequences until we achieved a good balance in quality. Figure 2.2 visualizes part of the 30 sequences for validation and testing of the REDS dataset.

Figure 2.6 shows an example of set of degraded images provided in REDS. The proposed REDS dataset is available online [2].

The REDS dataset was used to encourage development of video deblurring and super-resolution algorithms in NTIRE 2019 Challenges [77, 78]. REDS was further employed in NTIRE 2020 Challenge to promote image deblurring, application on mobile devices, and video deblurring [76]. In NTIRE 2021 Challenge, REDS was employed in developing image restoration algorithms for hybrid degradations such as low-resolution and JPEG artifacts jointly with motion blur [75].

[2] `https://seungjunnah.github.io/Datasets/reds`

2.4 Conclusion

In this chapter, we introduced the large-scale deblurring datasets for image and video deblurring. The GOPRO and the REDS datasets contain realistic blur as well as the ground truth sharp images that are temporally center-aligned with the pairs. They facilitate the training learning-based methods for deblurring and quantitative benchmarks of deblurring algorithms as the standard benchmark datasets.

Chapter 3

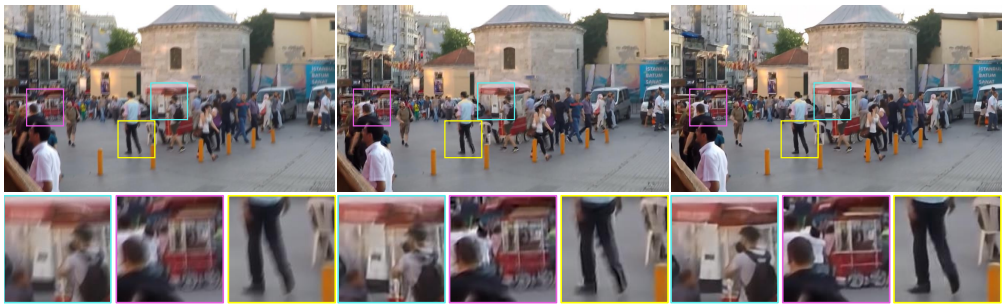
Deep Multi-Scale Convolutional Neural Networks for Single Image Deblurring

3.1 Introduction

The goal of single image deblurring problem is to estimate the unknown sharp image from a given blurry image. Earlier studies focused on handling the translational and rotational camera shakes and more later works tried to handle general object motions in dynamic environments. Most of these approaches are based on the blur model [111, 32, 37, 33] as (1.2). In practice, the blur kernel derived from the physical motion as well as the latent sharp image has to be jointly estimated from a single blurry image.

Due to the lack of blur datasets with ground truth sharp images, the previous approaches developed energy-minimization methods as (2.1). For example, [43] assumed locally uniform blur in image segments and [44] proposed segmentation-free method by considering the blur to be locally linear. However, such blur kernel approximation methods could be inaccuracey, especially in the cases of abrupt motion discontinuities and occlusions. In the joint energy minimization frameworks, erroneous kernel estimation affects the quality of the latent image, resulting in undesired ringing artifacts.

Recently, CNNs (Convolutional Neural Networks) have been applied in numer-



(a) (b) (c)

Figure 3.1: **(a) Input blurry image. (b) Result of Sun *et al.* [99]. (c) Our deblurred result.** Our results show clear object boundaries without artifacts.

ous computer vision problems including deblurring problem and showed promising results [115, 92, 99, 10]. Since no pairs of real blurry image and ground truth sharp image are available for supervised learning, they commonly used blurry images generated by convolving synthetic blur kernels. In [115, 92, 10], synthesized blur images with uniform blur kernel are used for training. And, in [99], classification CNN is trained to estimate locally linear blur kernels. Thus, CNN-based models are still suited only to some specific types of blurs, and there are restrictions on more common spatially varying blurs.

Therefore, the existing methods still have many problems before they could be generalized and used in practice. These are mainly due to the use of simple and unrealistic blur kernel models. Thus, to solve those problems, in this chapter, we propose a novel end-to-end deep learning approach for dynamic scene deblurring by using the GOPRO dataset proposed in chapter 2.

First, we propose a multi-scale CNN that directly restores latent images without assuming any restricted blur kernel model. Especially, the multi-scale architecture is designed to mimic conventional coarse-to-fine optimization methods. Unlike other approaches, our method does not estimate explicit blur kernels. Accordingly, our method is free from artifacts that arise from kernel estimation errors. Second, we train the pro-

posed model with a multi-scale loss that is appropriate for coarse-to-fine architecture that enhances convergence greatly. In addition, we further improve the results by employing adversarial loss [30]. Third, we propose a new realistic blurry image dataset with ground truth sharp images. To obtain kernel model-free dataset for training, we employ the dataset acquisition method introduced in [47]. As the blurring process can be modeled by the integration of sharp images during shutter time [47, 60, 45], we captured a sequence of sharp frames of a dynamic scene with a high-speed camera and averaged them to generate a blurry image by considering gamma correction.

By training with the GOPRO dataset and adding proper augmentation, our model can handle general local blur kernel implicitly. As the loss term optimizes the result to resemble the ground truth, it even restores occluded regions where blur kernel is extremely complex as shown in Figure 3.1. We trained our model with millions of pairs of image patches and achieved significant improvements in dynamic scene deblurring. Extensive experimental results demonstrate that the performance of the proposed method is far superior to those of the state-of-the-art dynamic scene deblurring methods in both qualitative and quantitative evaluations.

3.1.1 Related Works

There are several approaches that employed CNNs for deblurring [115, 99, 92, 10].

Xu *et al.* [115] proposed an image deconvolution CNN to deblur a blurry image in a non-blind setting. They built a network based on the separable kernel property that the (inverse) blur kernel can be decomposed into a small number of significant filters. Additionally, they incorporated the denoising network [24] to reduce visual artifacts such as noise and color saturation by concatenating the module at the end of their proposed network.

On the other hand, Schuler *et al.* [92] proposed a blind deblurring method with CNN. Their proposed network mimics conventional optimization-based deblurring methods and iterates the feature extraction, kernel estimation, and the latent image

estimation steps in a coarse-to-fine manner. To obtain pairs of sharp and blurry images for network training, they generated uniform blur kernels using a Gaussian process and synthesized lots of blurry images by convolving them to the sharp images collected from the ImageNet dataset [19]. However, they reported performance limits for large blurs due to their suboptimal architecture.

Similarly to the work of Couzinie-Devy *et al.* [16], Sun *et al.* [99] proposed a sequential deblurring approach. First, they generated pairs of blurry and sharp patches with 73 candidate blur kernels. Next, they trained classification CNN to measure the likelihood of a specific blur kernel of a local patch. And then smoothly varying blur kernel is obtained by optimizing an energy model that is composed of the CNN likelihoods and smoothness priors. Final latent image estimation is performed with conventional optimization method [127].

Note that all these methods require an accurate kernel estimation step for restoring the latent sharp image. In contrast, our proposed model is learned to produce the latent image directly without estimating blur kernels.

In other computer vision tasks, several forms of coarse-to-fine architecture or multi-scale architecture were applied [25, 23, 20, 66, 22]. However, not all multi-scale CNNs are designed to produce optimal results, similarly to [92]. In depth estimation, optical flow estimation, etc., networks usually produce outputs having smaller resolution compared to input image resolution [25, 23, 22]. These methods have difficulties in handling long-range dependency even if multi-scale architecture is used.

Therefore, we make a multi-scale architecture that preserves fine-grained detail information as well as long-range dependency from the coarser scales. Furthermore, we make sure intermediate level networks help the final stage in an explicit way by training network with multi-scale losses.

3.1.2 Kernel-Free Learning for Dynamic Scene Deblurring

Conventionally, it has been essential to find blur kernel before estimating latent image. CNN based methods were no exception [92, 99]. However, estimating kernel involves several problems. First, assuming simple kernel convolution cannot model several challenging cases such as occluded regions or depth variations. Second, kernel estimation process is subtle and sensitive to noise and saturation, unless blur model is carefully designed. Furthermore, incorrectly estimated kernels give rise to artifacts in latent images. Third, finding spatially varying kernel for every pixel in dynamic scene requires a huge amount of memory and computation.

Therefore, we adopt kernel-free methods in both the blur dataset and latent image estimation. In blurry image generation, we used GOPRO dataset to follow and approximate camera imaging process, rather than assuming specific motions, instead of finding or designing complex blur kernel. Note that the GOPRO dataset is composed of blurry and sharp image pairs only, and that the local kernel information is implicitly embedded in it. For latent image estimation, we do not assume blur sources and train the model solely on our blurry and sharp image pairs. Thus, our proposed method does not suffer from kernel-related problems in deblurring.

3.2 Proposed Method

In our model, finer scale image deblurring is aided by coarser scale features. To exploit coarse and middle level information while preserving fine level information at the same time, input and output to our network take the form of Gaussian pyramids. Note that most of other coarse-to-fine networks take a single image as input and output.

3.2.1 Model Architecture

In addition to the multi-scale architecture, we employ a slightly modified version of residual network structure [34] as a building block of our model. Using residual net-

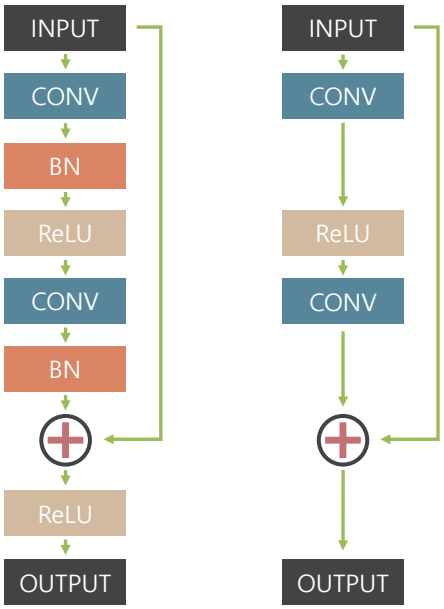


Figure 3.2: **(a) Original residual network building block. (b) Modified building block of our network.** We did not use batch normalization layers since we trained the models with mini-batch of size 2, which is smaller than usual for batch normalization. We found removing rectified linear unit just before the block output is beneficial in terms of performance, empirically.

work structure enables deeper architecture compared to a plain CNN. Also, as blurry and sharp image pairs are similar in values, it is efficient to let parameters learn the difference only. We found that removing the rectified linear unit after the shortcut connection of the original residual building block boosts the convergence speed at training time. We denote the modified building block as ResBlock. The original and our modified building block are compared in Figure 3.2.

By stacking enough number of convolution layers with ResBlocks, the receptive field at each scale is expanded. Details are described in the following paragraphs. For sake of consistency, we define scale levels in the order of decreasing resolution (i.e. level 1 for finest scale). Unless denoted otherwise, we use total $K = 3$ scales. At

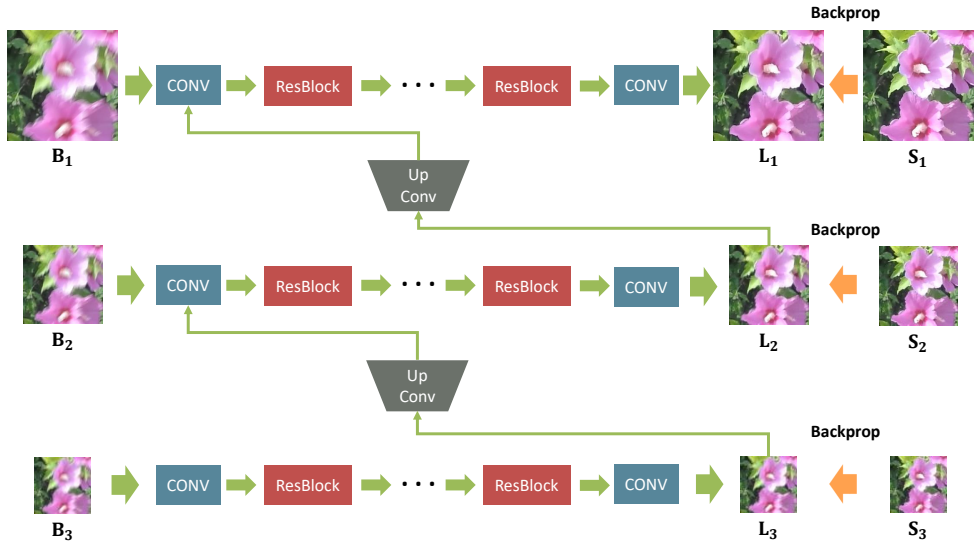


Figure 3.3: **The proposed multi-scale network architecture.** B_k, L_k, S_k denote the blurry, the latent, and the ground truth sharp images, respectively. Subscript k denotes the k-th scale level in the Gaussian pyramid, which is downsampled to $1/2^k$ scale. Our model takes a blurry image pyramid as the input and outputs an estimated latent image pyramid. Every intermediate scale output is trained to be sharp by multi-scale loss. At test time, the deblurred image at the original scale is chosen as the final result.

training time, we set the resolution of the input and output Gaussian pyramid patches to be $\{256 \times 256, 128 \times 128, 64 \times 64\}$. The scale ratio between consecutive scales is 0.5. For all convolution layers, we set the filter size to be 5×5. As our model is fully convolutional, at test time, the patch size may vary as the GPU memory allows. The overall architecture is shown in Figure 3.3.

Coarsest level network

At the front of the network locates the coarsest level network. The first convolution layer transforms 1/4 resolution, 64×64 size image into 64 feature maps. Then, 19 ResBlocks are stacked followed by last convolution layer that transforms the feature map into input dimension. Every convolution layer preserves resolution with zero padding. In total, there are 40 convolution layers. The number of convolution layers at each

scale level is determined so that total model should have 120 convolution layers. Thus, the coarsest level network has receptive field large enough to cover the whole patch. At the end of the stage, the coarsest level latent sharp image is generated. Moreover, information from the coarsest level output is delivered to the next stage where finer scale network is. To convert a coarsest output to fit the input size of the next finer scale, the output patch passes an upconvolution [64] layer, while other multi-scale methods use reshaping [25] or upsampling [20, 23, 66]. Since the sharp and blurry patches share low-frequency information, learning suitable feature with upconvolution helps to remove redundancy. In our experiment, using upconvolution showed better performance than upsampling. Then, the upconvolution feature is concatenated with the finer scale blurry patch as an input.

Finer level network

Finer level networks basically have the same structure as in the coarsest level network. However, the first convolution layer takes the sharp feature from the previous stage as well as its own blurry input image, in a concatenated form. Every convolution filter size is 5×5 with the same number of feature maps as in the coarsest level. Except for the last finest scale, there is an upconvolution layer before the next stage. At the finest scale, the original resolution sharp image is restored.

3.2.2 Training

Our model is trained on the proposed GOPRO dataset. Among 3214 pairs, 2103 pairs were used for training and remainings were used for the test. To prevent our network from overfitting, several data augmentation techniques are involved. In terms of geometric transformations, patches are randomly flipped horizontally and vertically, rotated by 90 degrees. For color, RGB channels are randomly permuted. To take image degradations into account, saturation in HSV colorspace is multiplied by a random number within $[0.5, 1.5]$. Also, Gaussian random noise is added to blurry images. To

make our network be robust against different strengths of noise, standard deviation of noise is also randomly sampled from Gaussian distribution, $N(0, (2/255)^2)$. Then, value outside [0, 1] is clipped. Finally, 0.5 is subtracted to set input and output value range zero-centered, having range [-0.5, 0.5].

In optimizing the network parameters, we trained the model in a combination of two losses, multi-scale content loss and adversarial loss.

Multi-scale content loss

Basically, the coarse-to-fine approach desires that every intermediate output becomes the sharp image of the corresponding scale. Thus, we train our network so that intermediate outputs should form a Gaussian pyramid of sharp images. MSE criterion is applied to every level of pyramids. Hence, the loss function is defined as

$$\mathcal{L}_{cont} = \frac{1}{2K} \sum_{k=1}^{K} \frac{1}{c_k w_k h_k} \|L_k - S_k\|^2, \tag{3.1}$$

where L_k, S_k denote the model output and ground truth image at scale level k, respectively. The loss at each scale is normalized by the number of channels c_k, width w_k, and the height h_k (i.e. the total number of elements).

Adversarial loss

Recently, adversarial networks are reported to generate sharp realistic images [30, 20, 88]. Following the architecture introduced in [88], we build discriminator as in Table 3.1. Discriminator takes the output of the finest scale or the ground truth sharp image as input and classifies if it is deblurred image or sharp image.

The adversarial loss is defined as

$$\mathcal{L}_{adv} = \mathop{\mathbb{E}}_{S \sim p_{sharp}(S)} [\log D(S)] + \mathop{\mathbb{E}}_{B \sim p_{blurry}(B)} [\log(1 - D(G(B)))], \tag{3.2}$$

#	Layer	Weight dimension	Stride
1	conv	$32 \times 3 \times 5 \times 5$	2
2	conv	$64 \times 32 \times 5 \times 5$	1
3	conv	$64 \times 64 \times 5 \times 5$	2
4	conv	$128 \times 64 \times 5 \times 5$	1
5	conv	$128 \times 128 \times 5 \times 5$	4
6	conv	$256 \times 128 \times 5 \times 5$	1
7	conv	$256 \times 256 \times 5 \times 5$	4
8	conv	$512 \times 256 \times 5 \times 5$	1
9	conv	$512 \times 512 \times 4 \times 4$	4
10	fc	$512 \times 1 \times 1 \times 1$	-
11	sigmoid	-	-

Table 3.1: **Architecture details of the discriminator model.** Every convolutional layer is activated by the following LeakyReLU layer.

where *G* and *D* denote the generator, that is our multi-scale deblurring network in Figure 3.3 and the discriminator (classifier), respectively. When training, *G* tries to minimize the adversarial loss while *D* tries to maximize it.

Finally, by combining the multi-scale content loss and adversarial loss, the generator network and discriminator network is jointly trained. Thus, our final loss term is

$$\mathcal{L}_{total} = \mathcal{L}_{cont} + \lambda \times \mathcal{L}_{adv}, \tag{3.3}$$

where the weight constant $\lambda = 1 \times 10^{-4}$.

We used ADAM [49] optimizer with a mini-batch size 2 for training. The learning rate is adaptively tuned beginning from 5×10^{-5}. After 3×10^5 iterations, the learning rate is decreased to 1/10 of the previous learning rate. Total training takes 9×10^5 iterations to converge.

3.3 Experiments

We implemented our model with torch7 library. All the following experiments were performed in a desktop with i7-6700K CPU and NVIDIA GTX Titan X (Maxwell) GPU.

3.3.1 Comparison on GOPRO Dataset

We evaluate the performance of our model in the proposed GOPRO dataset. Our test dataset consists of 1111 pairs, which is approximately $1/3$ of the total dataset. We compare the results with those of the state-of-the-art methods [44, 99] in both qualitative and quantitative ways. Our results show significant improvement in terms of image quality. Some deblurring results are shown in Figure 3.4. We notice from the results of Sun *et al.* [99], deblurring is not successful on the regions where blurs are nonlinearly shaped or located at the boundary of motion. Kim and Lee [44]'s results also fail in cases where strong edges are not found. In contrast, our results are free from those kernel-estimation related problems. Table 3.2 shows the quantitative evaluation results of the competing methods and ours with different scale level k in terms of PSNR, SSIM over the test data. Also, the runtime is compared. We observe that our system with $K = 2$ produces the best results in terms of both PSNR and SSIM, while $K = 3$ is the fastest. In Figure 3.5 and 3.6, more comparisons on different scenes are shown.

Metric	[99]	[44]	Ours		
			$K = 1$	$K = 2$	$K = 3$
PSNR	24.64	23.64	28.93	**29.23**	29.08
SSIM	0.8429	0.8239	0.9100	**0.9162**	0.9135
Runtime	20 min	1 hr	7.21 s	4.33 s	**3.09 s**

Table 3.2: **Quantitative deblurring performance and running time comparison on GOPRO dataset [72].** K denotes the number of scale levels.

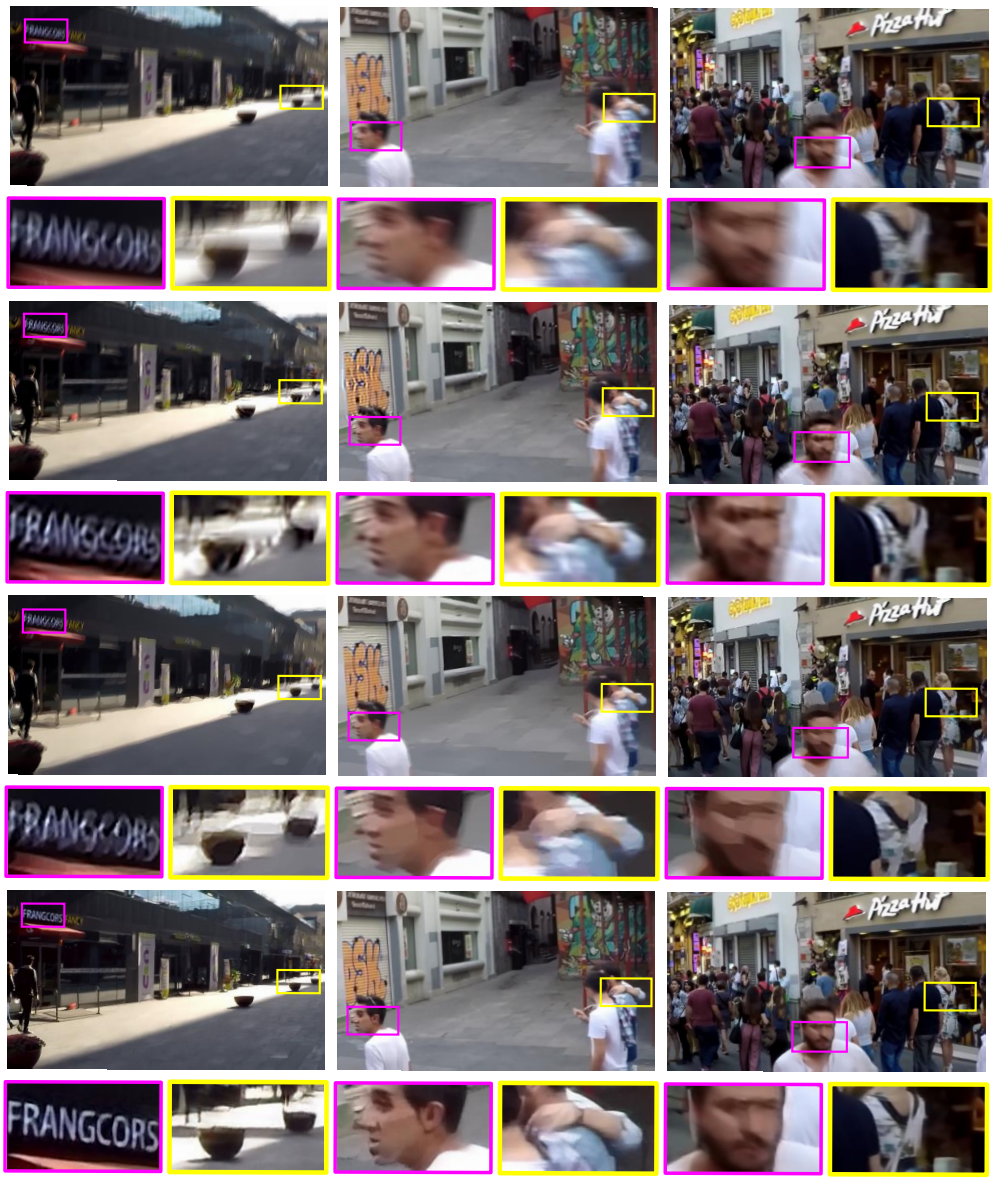


Figure 3.4: **Test results on the GOPRO dataset [72].** From top to bottom: Blurry images, results of Sun *et al.* [99], results of Kim and Lee [44], and results of the proposed method.

Figure 3.5: **Visual comparison with other methods on GOPRO dataset [72].**

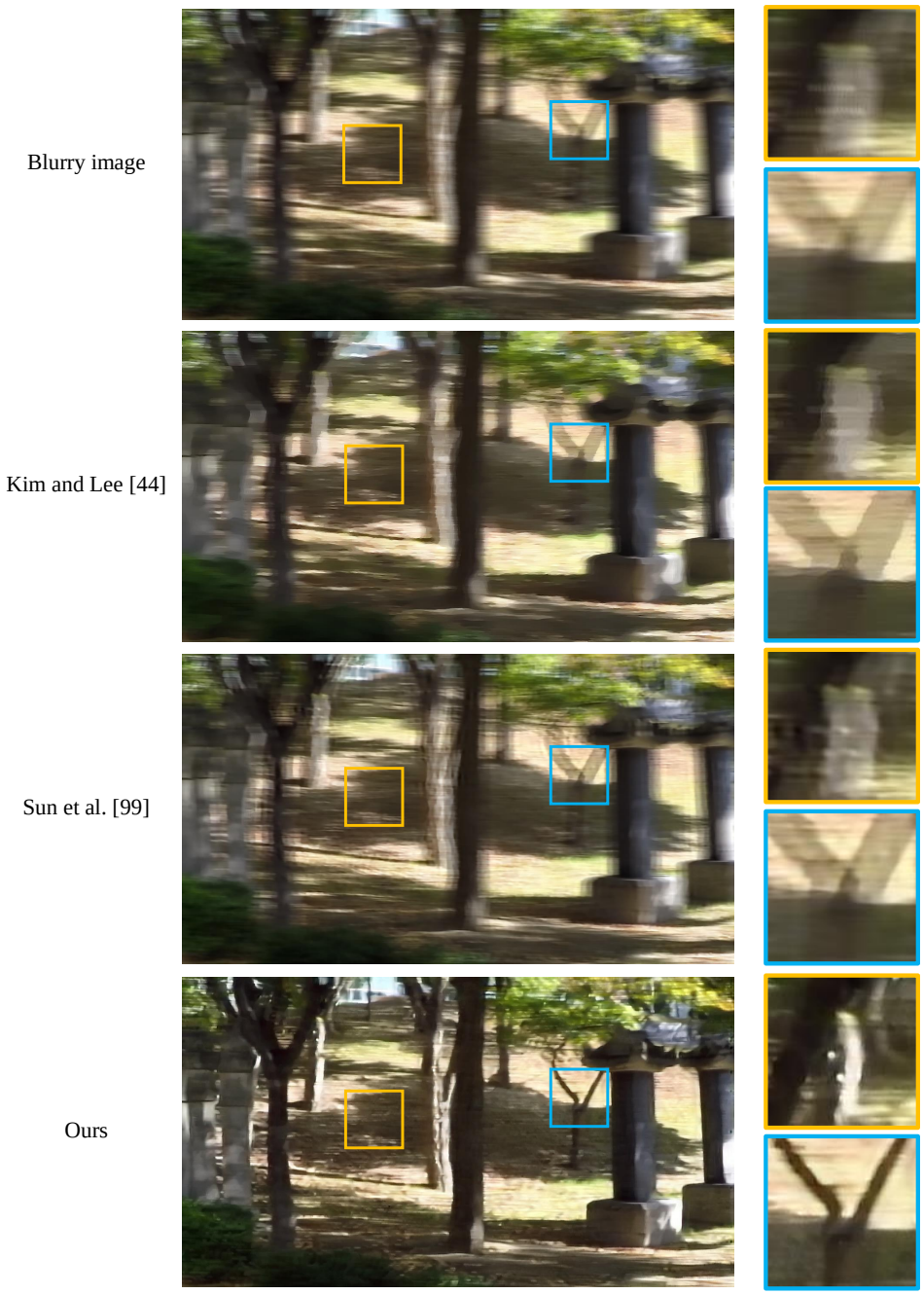


Figure 3.6: **Visual comparison with other methods on GOPRO dataset [72].**

3.3.2 Comparison on Köhler Dataset

Köhler dataset [50] consists of 4 latent images and 12 differently blurred images for each of them. The blurs are caused by replaying recorded 6D camera motion, assuming linear CRF. We report the quantitative results on this dataset in Table 3.3. For compatibility with the Köhler dataset, our model is trained by setting CRF as identity function in GOPRO dataset. We note that our system with $K = 2$ produces the best results in PSNR, and the system $K = 3$ exhibits the best MSSIM result.

Metric	[99]	[44]	Ours		
			$K = 1$	$K = 2$	$K = 3$
PSNR	25.22	24.68	25.74	26.02	26.48
MSSIM	0.7735	0.7937	0.8042	0.8116	0.8079

Table 3.3: **Quantitative comparison on Köhler dataset [50]**. The dataset has its own evaluation code, thus we report multi-scale SSIM instead of SSIM.

3.3.3 Comparison on Lai *et al.* [54] dataset

Lai *et al.* [54] generated synthetic dataset by convolving nonuniform blur kernels and imposing several common degradations. They also recorded 6D camera trajectories to generate blur kernels. However, their blurry images and sharp images are not temporally center-aligned as GOPRO dataset, making reference-based image quality metrics such as PSNR and SSIM to be less correlated with the perceptual quality. Thus, we show qualitative comparisons in Figure 3.7. Clearly, our results avoid ringing artifacts while preserving details such as wave ripple.

In [54], the dataset also includes real blurry images. We present qualitative comparisons of deblurred results in Figure 3.8 and 3.9.

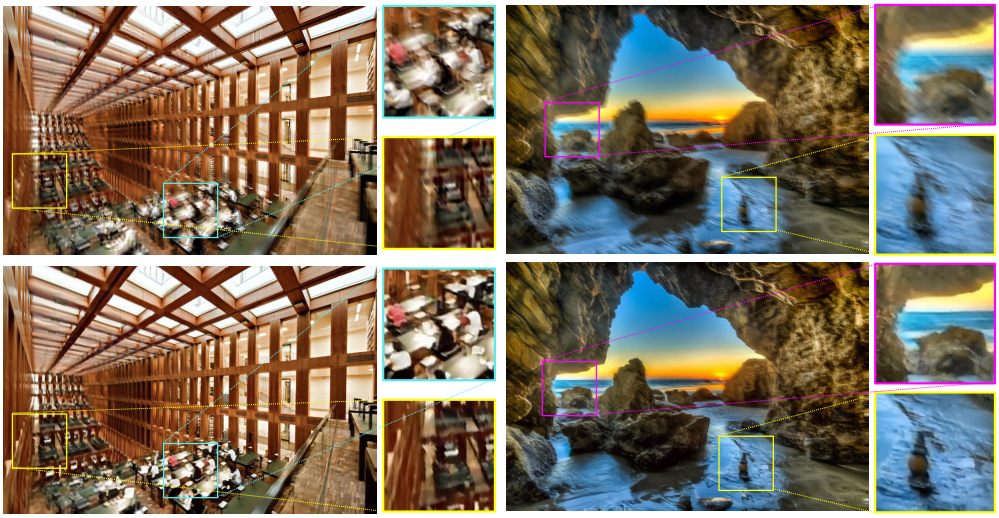


Figure 3.7: **Visual comparison of the deblurred results on Lai *et al.* dataset [54].** The top row shows results of Sun *et al.* [99] and the bottom row shows our results.

3.3.4 Comparison on Real Dynamic Scenes

Finally, we further present deblurring results on real dynamic scenes. The blurry scenes are captured by a SONY RX100 M4 camera. The qualitative deblurring results of Kim and Lee [44], Sun *et al.* [99] and ours are compared in Figure 3.10 and 3.11.

3.3.5 Effect of Adversarial Loss

In training our proposed models, we combined both the multi-scale content loss (MSE) and the adversarial loss. We examine the effect of the adversarial loss term quantitatively and qualitatively. The PSNR and SSIM results from the training with and without the adversarial loss are shown in table 3.4. From this results, we observe that adding adversarial loss does not increases PSNR, but increases SSIM, which means that it encourages to generate more natural and structure preserving images. Figure 3.12 and 3.13 show some qualitative comparisons between the results of our network trained with $\mathcal{L}_{cont}$ and $\mathcal{L}_{cont} + \lambda\mathcal{L}_{adv}$.

Figure 3.8: **Visual comparison with other methods on a real image in Lai *et al.* dataset [54].**

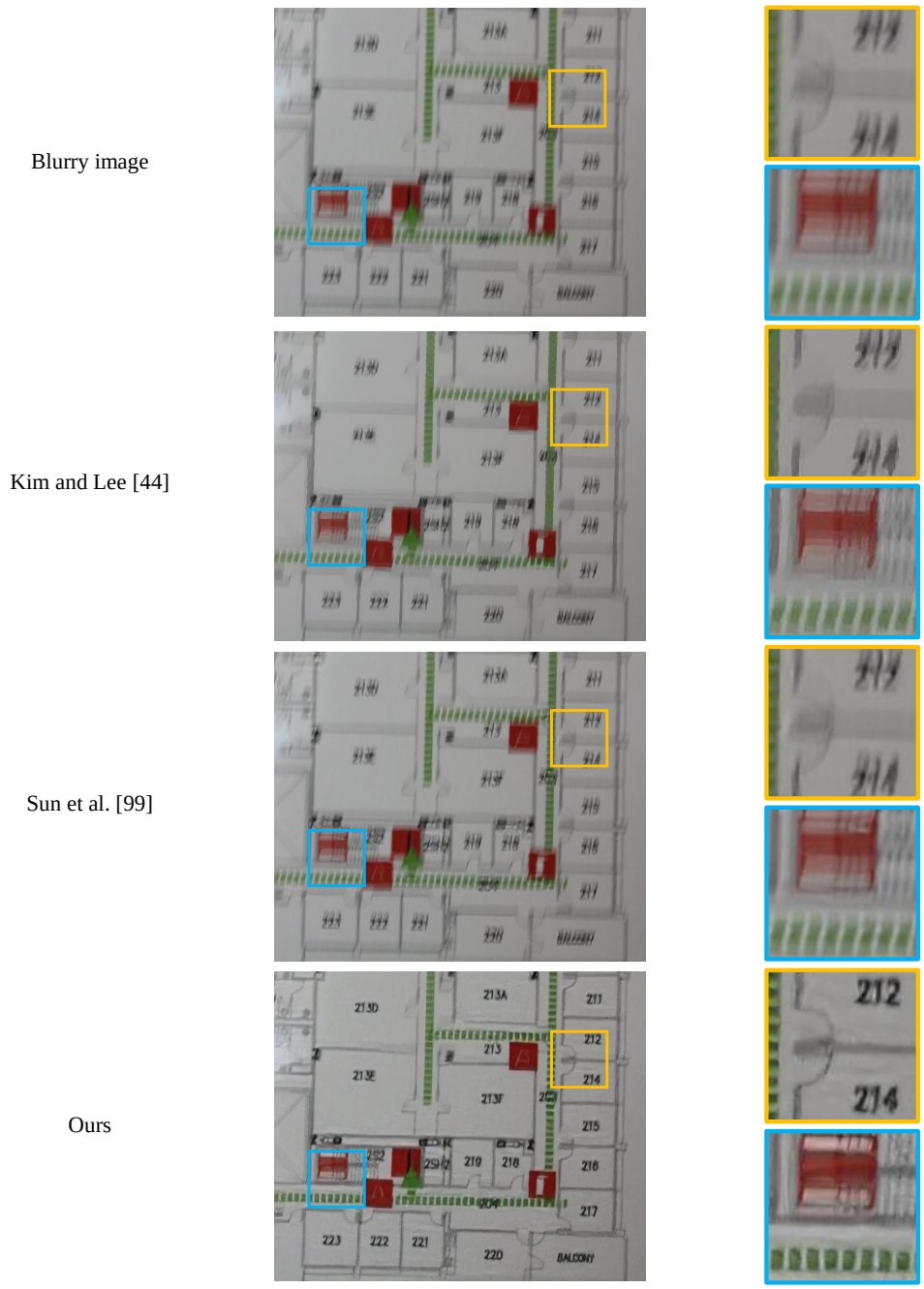


Figure 3.9: **Visual comparison with other methods on a real image in Lai *et al.* dataset [54].**

Figure 3.10: **Visual comparison with other methods on a real dynamic scene.**

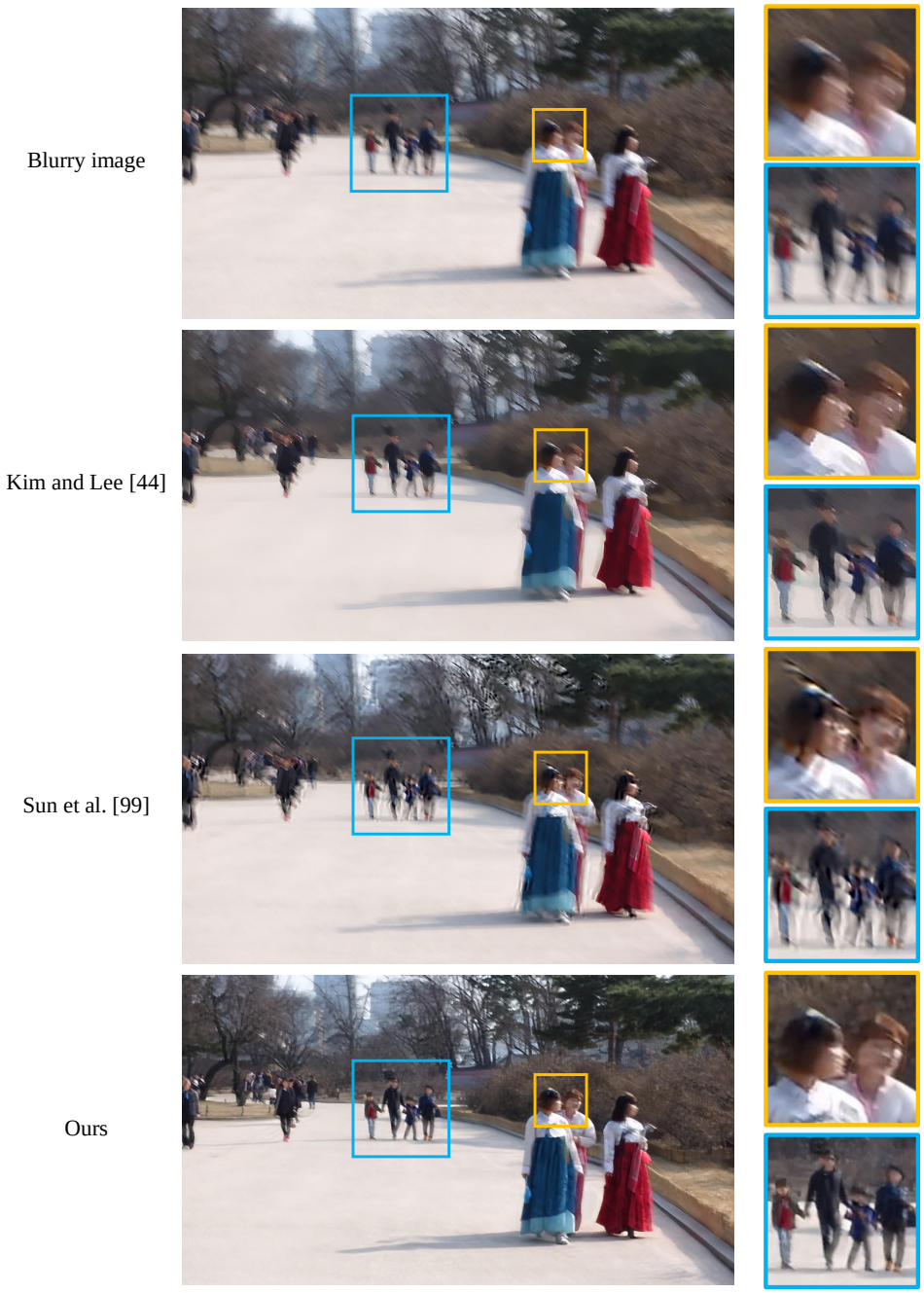
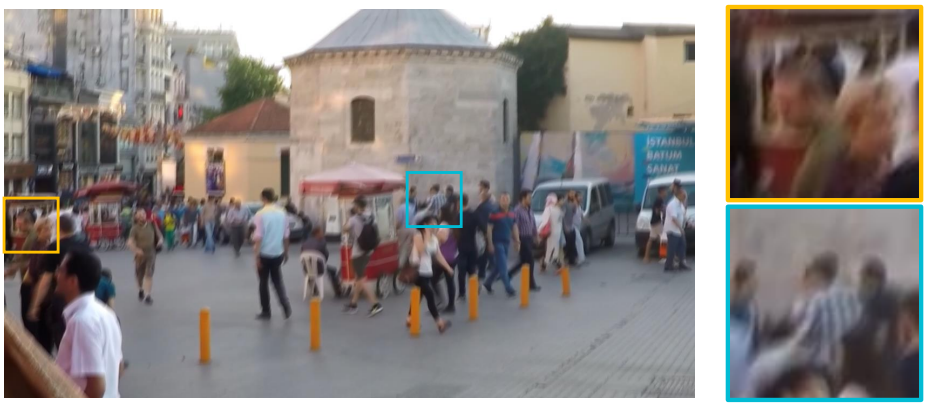


Figure 3.11: **Visual comparison with other methods on a real dynamic scene.**

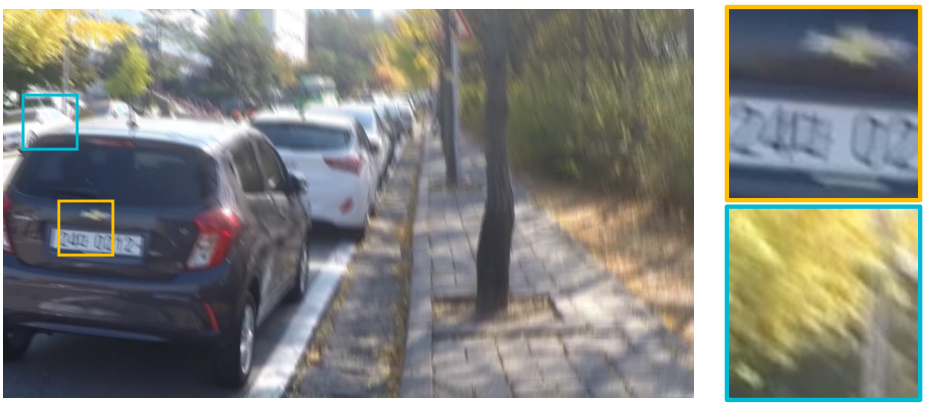


Blurry image

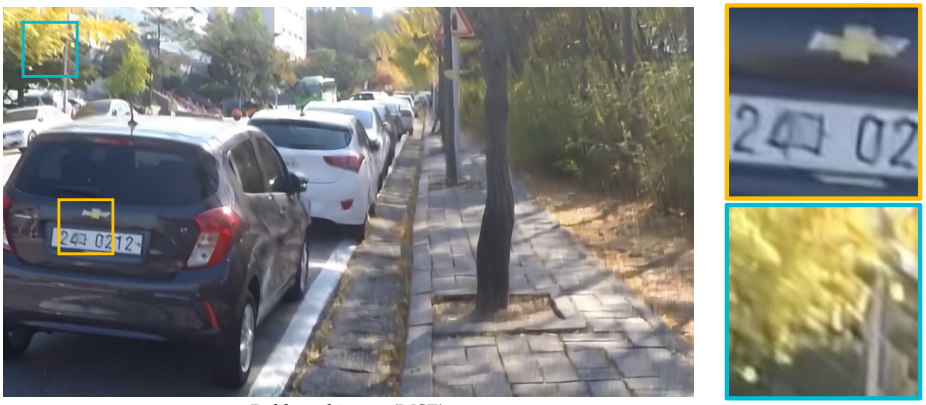
Deblurred image (MSE)

Deblurred image (MSE + Adversarial)

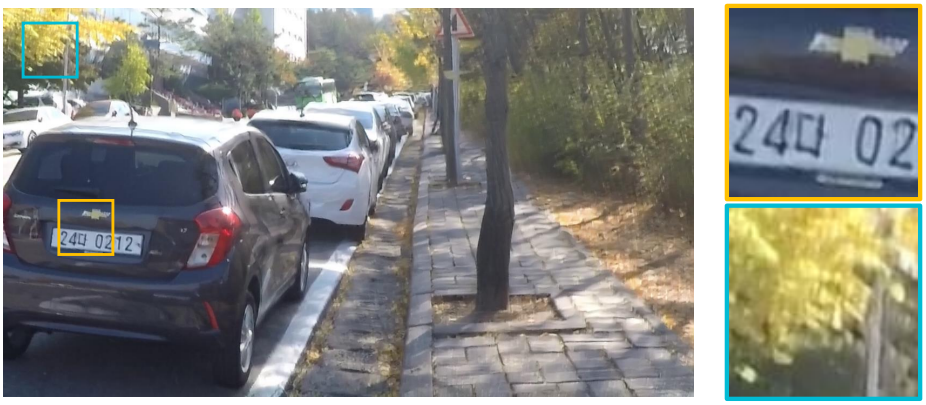
Figure 3.12: **Visual comparison of results from our model trained with different loss functions.** The blurry image is from GOPRO dataset [72].



Blurry image



Deblurred image (MSE)



Deblurred image (MSE + Adversarial)

Figure 3.13: **Visual comparison of results from our model trained with different loss functions.** The blurry image is from GOPRO dataset [72].

Loss	$\mathcal{L}_{cont}(MSE)$	$\mathcal{L}_{cont} + \lambda\mathcal{L}_{adv}$
PSNR	28.62	28.45
SSIM	0.9094	0.9170

Table 3.4: **Quantitative deblurring performance comparison by the loss function used to optimize our model ($K = 3,\ \lambda = 1 \times 10^{-4}$).** Evaluated on the GOPRO test dataset assuming linear CRF.

3.4 Conclusion

In this chapter, we proposed a blind deblurring neural network for sharp image estimation. Unlike previous studies, our model avoids problems related to kernel estimation. The proposed model follows a coarse-to-fine approach and is trained in multi-scale space. Experimental results show that our approach outperforms the state-of-the-art methods in both qualitative and quantitative ways while being much faster.

Chapter 4

Intra-Frame Iterative RNNs for Video Deblurring

4.1 Introduction

In contrast to the photographs that are meant to capture a single moment, videos are essentially captured in dynamic environments with temporally varying contents. Cameras are typically hand-held and are more likely to be shaken during shooting, and fast moving objects can abruptly occur at any time. Considering the practical needs in video recordings, we aim to develop a video deblurring method in addition to the single image deblurring method presented in chapter 3.

In video deblurring, it is crucial to analyze the relevant information between consecutive frames as well as the information in the target frame. In recent deep neural network based approaches, several designs of CNNs and RNNs are adopted to incorporate temporal information. Su et al. [97] introduced a 2-stage approach to handle misplacement from large motions between frames and the fuse information between the frames. A sequence of frames is spatially aligned to the middle frame by homography or optical flow. Those frames are then fed into a CNN to get a deblurred middle frame. On the other hand, Wieschollek et al. [112] and Kim et al. [46] proposed recurrent network architectures that can operate on arbitrary length videos. While [112] used information from past frames by simply copying features, [46] presented a Dy-

(a) Input blurry image $\mathbf{B}_t$ (b) Deblurred image $\mathbf{L}_t$, IFI-RNN (C2H3-reg)

(c) Blurry (d) RDN [112] (e) DBN+OF [97]

(f) OVD [46] (g) IFI-RNN (C2H1) (h) IFI-RNN (C2H3-reg)

Figure 4.1: **Deblurred result comparison with state-of-the-art methods.** (g) Result of our model with dual RNN cells without iteration. (h) Result of our 3-iteration model with stochastic regularization.

namic Temporal Blending module on a fast RNN. The module blends the hidden state from past frames and feature from the current frame to transfer the temporal information through hidden states.

These neural network-based approaches mainly focus on how to adopt the related information from the neighboring frames to restore the target frame and show significant improvements. However, these methods try to handle temporal relation in a single-step operation, which may not be optimal. Traditionally, the difficulty of estimating motion information or blur kernel from multiple frames was mitigated by iterative estimation steps [121, 114, 45, 1, 84]. Furthermore, handling the neighbor

frames with the alignment from optical flow [97] or heavy neural network [112] is expensive in computation. Thus, to resolve these issues, temporal information transfer method both fast and more optimal is required.

We set up a baseline model in a light and fast convolutional RNN architecture that exploits the inter-frame information. Like [46], we deliver information from past frames to the current frame in the form of hidden state. To let the propagated hidden state fit to the target frame, we employ an iterative hidden state update scheme within a single inter-frame time-step. We refer to this operation as intra-frame iterations. As the intra-frame iteration is in the same form as inter-frame operation, no modification of the architecture or additional parameters are required. Additionally, we investigate and analyze the schemes of intra-frame recurrence by varying the composition of the RNN cells. (i.e., single-cell and dual-cell methods) We experimentally show that the proposed intra-frame recurrence scheme results in substantial improvements in the restoration accuracy.

We train each model with a predefined intra-frame iteration number. On average, more iterations bring performance improvements. However, not all frames are best restored from the maximum number of iterations. As this is the case when more computation induces degradation, we cast this as an imperfect optimization issue. We adopt a stochastic strategy [106] to employ regularization effect to improve iterative models. As the models with different iteration numbers share an architecture, we regard less iteration models as parts of larger iteration models. During training, the number of internal iterations is chosen randomly. However, our regularization loss term favors fewer calculations. Several works have been reported that training a model with partial computation paths at random improves accuracy [96, 108, 38, 31, 106]. We implement the training by using a gating unit that decides iteration numbers. Note that our primary goal is to improve performance by regularizing RNN cells. Therefore, we drop the gating function at inference and prevent the model from showing a stochastic or adaptive behavior. The result of our regularized dual-cell method is displayed in Figure 4.1.

Our contributions in this chapter is summarized as follows:

- We present a simple yet effective RNN-based video deblurring method that exploits both the intra-frame (internal) and inter-frame (external) recurrent schemes. By updating the hidden state multiple times internally during a single time-step, our model produces better results without modifying the architecture.
- We study various types of intra-frame iteration strategies. For recurrent networks with different internal cell parameters, we investigate the effect of partial recurrence to investigate more optimal hidden state update strategy.
- Finally, we develop a single model that can be trained to handle various internal recurrence paths (iterations). Our loss function is composed of a data term that aims to minimize restoration error and a prior term that favors shorter computation path. We train our multi-path network in a stochastic way. Owing to the regularization effect of the stochastic training that prevents the co-adaptation of layers, the flexible intra-frame iterative model provides more improved deblurring results.
- Through extensive empirical tests and evaluations, we demonstrate the superiority of the proposed model over the current state-of-the-art methods in both deblurring accuracy and computational efficiency.

4.2 Related Works

In this section, we describe previous works related to the proposed method.

Video Deblurring

In the early studies of video deblurring, the concept of lucky imaging was adopted where sharp contents replaced the blurry ones in pixel [67] and patch [15] level. Later, deconvolution based methods were widely studied where kernels are estimated from

inter-frame relation. Temporal information was exploited to predict the global motion and to generate a sharp panorama scene from a blurry video [60]. To handle differently blurred regions, Wulff and Black [114] studied layered blur model that segments an image into layers and deconvolved each layer separately to improve the estimation of both the blur kernels and the latent image. Kim et al. [45, 47] proposed a segmentation-free dynamic video deblurring method where locally varying blur kernels were approximated from bidirectional optical flows. These methods formulate the problem as a non-convex energy minimization framework of which variables include the local blur kernels and the latent images. Thus, many deconvolution algorithms for deblurring [43, 44, 121, 45, 99] resolve this issue by iteratively optimizing the energy function.

Recently, [72, 97] introduced video datasets that contain realistic blurry frames and corresponding sharp ground-truth frames. As the frames in a video recorded by a high-speed camera are sharp and slowly changing, the average of several subsequent frames can mimic a blurry frame captured at a longer exposure. With the advent of realistic blur datasets, there have been proposed a few deep learning-based methods for single image [72] and video [97] deblurring. Similarly, Wieschollek et al. [112] synthesized the training data by downsampling and interpolating 4k-8k resolution videos.

Su et al. [97] proposed a CNN-based algorithm called DBN. It takes a stack of 5 successive frames as an input and deblurs the middle frame among them. To handle severely blurred frames, they also aligned their input frames with the optical flow as a pre-processing. On the other hand, RDN [112] uses an encoder-decoder architecture model that can process arbitrary length videos. RDN utilizes temporal skip connections so that features extracted in the previous frames can directly propagate to the next frame. In advance, OVD [46] proposed a recurrent network whose hidden state carries the temporal information from the past time-steps. In the recurrent architecture, they added a dynamic temporal blending module so that the hidden state from the previous time-step is adapted to the current frame. Furthermore, Spatio-temporal Transformer

Network [48] was applied to improve DBN and OVD by making use of long-range pixel correspondences.

In this chapter, we aim to improve the deblurring quality using recurrent neural networks by updating the hidden states to be more optimal for predicting the output. In the viewpoint of making better use of hidden states, our work is closely related to [46, 48]. However, we reuse existing parameters without introducing any extra module.

Burst Deblurring

Under low-light conditions, a burst of photographs is likely to be blurred due to hand tremor. In [120, 9] the sparse prior of blur kernels and spatial gradient of latent images are investigated to obtain sharp images. On the other hand, some alignment-free methods were suggested by posing a joint problem of multiple image registration and deblurring [118, 12, 122].

Then, Delbracio and Sapiro [17, 18] presented a simple yet efficient burst deblurring method without relying on kernel estimation and deconvolution. They utilized spectral information in the Fourier domain where information from less blurred images is more weighted. Wieschollek et al. [113] further extended [17] by learning a hybrid network that decides the weights for Fourier burst accumulation and the deconvolution filter. Furthermore, a recently proposed permutation-invariant model by Aittala and Durand [2] improved the restoration quality significantly in the presence of noise, blur, and saturation. We also augment noise in the training process like [2, 72].

Stochastic Neural Network Training

Most of the neural networks are designed to process equally for every input. However, training the networks as it is not always known to be optimal. Therefore, several randomized training strategies have been proposed to regularize the optimization process. The most classical types of stochastic regularization techniques are Dropout [36, 96] and DropConnect [108]. While Dropout randomly deactivates the outputs of fully-

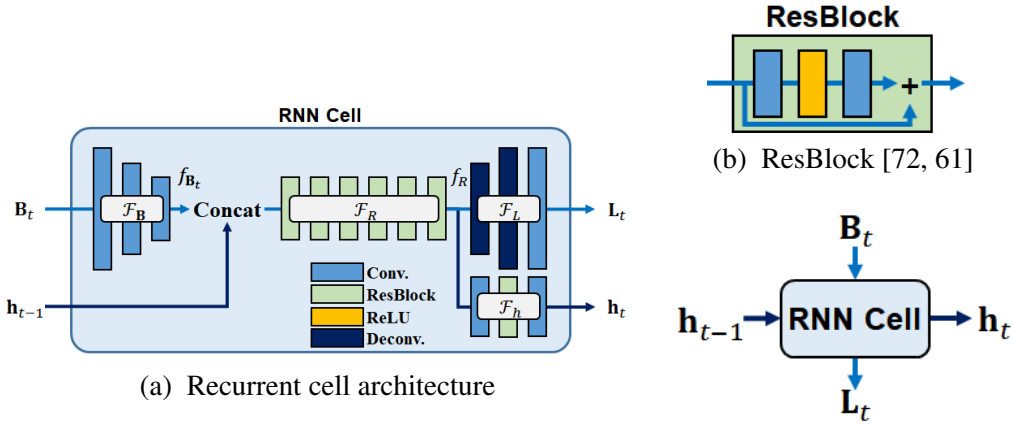
connected layers, DropConnect disconnects the weights of the layer at training time. They are known to prevent co-adaptation of features and regularize the network to avoid overfitting.

In ResNets [34, 35], residual blocks contain shortcut connections where their inputs are directly headed to the output in parallel with convolution features. Veit et al. [107] observed that this could be interpreted as an ensemble of exponentially many shallower networks. In ResNets, surprisingly, removing or permuting several layers do not cause catastrophic degradation. Furthermore, ResNets trained with random skips of residual blocks showed ameliorated classification accuracy [38]. Similarly, FractalNet [55] showed that drop-path training could also exhibit regularization effect.

Recently, more advanced stochastic training techniques were proposed, letting the stochastic path to be chosen by the model itself. Graves [31] proposed an adaptive computation time (ACT) algorithm where the number of recurrence steps between the inputs is decided by the network with an estimated halting score, instead of using a predefined fixed number of iterations. Figurnov et al. [27] extended the ACT to spatial locations of ResNets [35] so that every pixel would have different network depth.

The most relevant study to ours is the work by Veit and Belonge [106]. They added a gating unit in each block of the ResNet that could switch-off rather irrelevant layers. To computationally benefit from the switching, the output from the gates should be hard binary rather than being soft. The training of the hard gate is enabled by using the back-propagation with Gumbel-SoftMax relaxation [39, 65]. In contrast to the previous methods focusing on acceleration with a moderate increase in error, they exhibit improved accuracy compared to the original ResNet for image classification.

In our experiments, we find that several different numbers of intra-frame iterations are beneficial in general. Hence, we conjecture that training a single generic model that could operate in the variable number of intra-frame iterations is possible, regarding the shared architecture between our models. We aim to benefit from regularization effect through training our model in stochastic paths. To let our model decide the iteration



(a) Recurrent cell architecture

(b) ResBlock [72, 61]

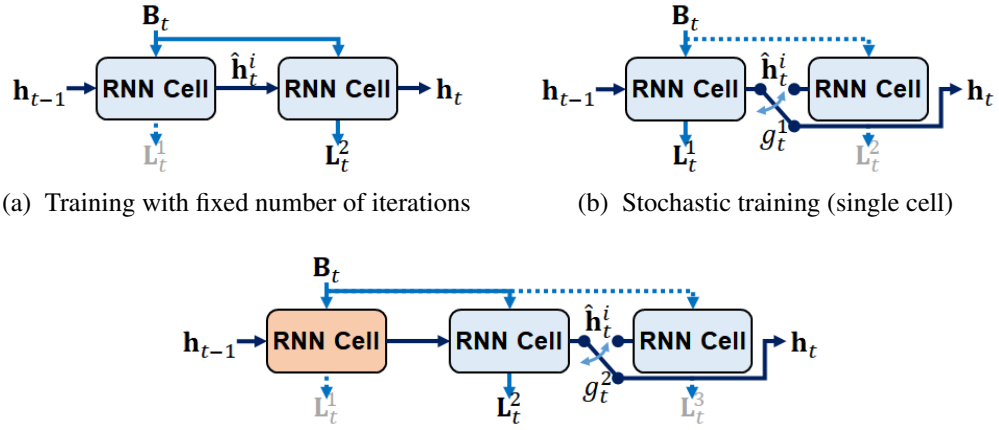
(c) RNN at t-th frame

Figure 4.2: **The baseline architecture of the proposed IFI-RNN**

number itself, we implement a stochastic gate function that determines if additional iteration is to be used or not. To jointly train the gate as well as the main network, we design a regularization loss term that favors less computation together with the content (L2) loss. We adopt the Gumbel-Softmax trick [39, 65] that has been used in [106] to route the model in a single prediction path that is discretely decided from the iteration number. Our regularized models exceed their original models in deblurring performance, both quantitatively and qualitatively.

4.3 Proposed Method

In this section, we describe how we develop our model. In section 4.3.1, we describe our baseline RNN model and the formulation terminologies. In section 4.3.2, we explain the concept of our intra-frame iteration model and analyze possible iteration strategies. Lastly, we describe more advanced training methods for our intra-frame iteration RNN in section 4.3.3.



(a) Training with fixed number of iterations

(b) Stochastic training (single cell)

(c) Stochastic training (dual cell)

Figure 4.3: **Training of IFI-RNN using different hidden state update schemes.**

4.3.1 Recurrent Video Deblurring Networks

Let us denote the blurry video, ground-truth sharp video, and the predicted latent video as $\mathbf{B} = \{\mathbf{B}_t\}$, $\mathbf{S} = \{\mathbf{S}_t\}$, $\mathbf{L} = \{\mathbf{L}_t\}$ with the frame index $t \in \{1 \dots T\}$, respectively.

We construct our baseline architecture as a recurrent neural network so that temporal information can propagate over video frames like [46]. Then, our network operates on the blurry input video by following recurrence operation.

$$(\mathbf{L}_t, \mathbf{h}_t) = \mathcal{F}(\mathbf{B}_t, \mathbf{h}_{t-1}),$$

where $\mathcal{F}$ refers to our RNN cell. The cell consists of several components, $\mathcal{F}_\mathbf{B}$, $\mathcal{F}_\mathbf{R}$, $\mathcal{F}_\mathbf{L}$, $\mathcal{F}_\mathbf{h}$ as shown in Figure 4.2. First, $\mathcal{F}_\mathbf{B}$ extracts the feature $f_{\mathbf{B}_t}$ from a blurry frame. Then, $\mathcal{F}_\mathbf{R}$ produces the intermediate feature $f_{\mathbf{B}_t}$ that is used for $\mathcal{F}_\mathbf{L}$ and $\mathcal{F}_\mathbf{h}$ to estimate the latent frame $\mathbf{L}_t$ and hidden state $\mathbf{h}_t$, respectively. $\mathbf{h}_t$ is the hidden state that is produced at t-th time-step and will be propagated to $t+1$-th time-step. We initialize $\mathbf{h}_0$ with zero.

The RNN cell consists of strided convolutions ($\mathcal{F}_\mathbf{B}$) followed by ResBlocks [72, 61] without batch normalization ($\mathcal{F}_\mathbf{R}$, $\mathcal{F}_\mathbf{h}$), and up-convolutions ($\mathcal{F}_\mathbf{L}$).

We train our baseline model with the $L2$ loss between the estimated latent video and the ground-truth sharp video as

$$\mathcal{L}_{\text{content}} = \frac{1}{TCHW} \sum_{t=1}^{T} \|\mathbf{L}_t - \mathbf{S}_t\|_2^2, \tag{4.1}$$

where C, H, W denote the number of channels (3 for RGB color videos), height, and the width of the training samples, respectively.

4.3.2 Intra-Frame Iteration Model

The most crucial part of RNNs against CNNs is the hidden state that brings the performance gain as CNNs have no temporal connections. Therefore, it is essential to have *good* hidden states so that they could better help to predict more accurate outputs at the current frame as well as at the next frame. In this regard, we attempt to make *better* use of hidden states by intra-frame iteration before passing it to the next RNN cell.

We implement this idea by utilizing our baseline RNN cell architecture. First, we compute the initial hidden state $\hat{\mathbf{h}}_t^0$ at a certain time step t from a blurry input $\mathbf{B}_t$ and the previous hidden state $\mathbf{h}_{t-1}$ using our RNN cell. Then, we feedback $\hat{\mathbf{h}}_t^0$ to the cell again without changing $\mathbf{B}_t$ to update the hidden state. After updating the hidden state for N iterations, we finally generate a latent output frame $\mathbf{L}_t$ at that time step with the updated hidden state $\hat{\mathbf{h}}_t^N$. Note that the blur feature extractor $\mathcal{F}_{\mathbf{B}}$ and the latent frame estimator $\mathcal{F}_{\mathbf{L}}$ are used only once despite the number of iterations.

The blur feature extraction part, $\mathcal{F}_{\mathbf{B}}$, contains three convolutional layers without nonlinear activations. It reduces the spatial resolution and effectively increases the receptive field. Given an RGB input size $h \times w$, it produces feature size $60 \times h/4 \times w/4$. Then, the extracted feature is concatenated with a hidden state of size $20 \times h/4 \times w/4$, producing a tensor of shape $80 \times h/4 \times w/4$.

$\mathcal{F}_{\mathbf{R}}$ is a sequence of the following 6 residual blocks. It generates a feature map having the same size as its input, $80 \times h/4 \times w/4$. Each resblock consists of two

convolutional layers with ReLU activation in between. Note that there are no batch-normalization layers in the ResBlocks, following previous image deblurring [72] and super-resolution [61] models.

With the calculated feature from $\mathcal{F}_{\mathbf{R}}$, $\mathcal{F}_{\mathbf{h}}$ and $\mathcal{F}_{\mathbf{L}}$ are located in parallel. Each of them outputs the hidden state and the latent deblurred frame. $\mathcal{F}_{\mathbf{h}}$ has 2 convolutional layers with a single ResBlock in between. It preserves the resolution of the feature to generate the hidden state of the next time-step. On the other hand, $\mathcal{F}_{\mathbf{L}}$ increases the resolution and reduces the channels to reconstruct a deblurred frame. It uses a convolution layer after two up-convolutions. In Table 4.1, we explain the exact kernels and outputs of each layer.

Module	layer	kernel	stride	output size	#parameters
	input	-	-	$3 \times h \times w$	-
$\mathcal{F}_{\mathbf{B}}$	conv	5×5	1	$20 \times h \times w$	1520
	conv	5×5	2	$40 \times h/2 \times w/2$	20040
	conv	5×5	2	$60 \times h/4 \times w/4$	60060
	hidden state	-	-	$20 \times h/4 \times w/4$	-
	concat	-	-	$80 \times h/4 \times w/4$	-
$\mathcal{F}_{\mathbf{R}}$	ResBlock $\times 6$	3×3	1	$80 \times h/4 \times w/4$	692160
$\mathcal{F}_{\mathbf{h}}$	conv	3×3	1	$20 \times h/4 \times w/4$	14420
	ResBlock	3×3	1	$20 \times h/4 \times w/4$	7240
	conv	3×3	1	$20 \times h/4 \times w/4$	3620
$\mathcal{F}_{\mathbf{L}}$	up-conv	3×3	2	$40 \times h/2 \times w/2$	28840
	up-conv	3×3	2	$20 \times h \times w$	7220
	conv	5×5	1	$3 \times h \times w$	1503

Table 4.1: Our IFI-RNN cell architecture details. Each component details are shown. $\mathcal{F}_{\mathbf{B}}$, $\mathcal{F}_{\mathbf{R}}$, $\mathcal{F}_{\mathbf{L}}$, and $\mathcal{F}_{\mathbf{h}}$ each has 0.08M, 0.69M, 0.03M, 0.04M parameters, respectively. There are total 0.84M parameters in our single cell model.

Thus, our single cell method has 0.84M parameters. For dual cell method, we do not need $\mathcal{F}_{\mathbf{L}}$ for the 1st cell, as we estimate the latent image in the 2nd cell only.

Hence, our dual cell method only requires storage for 1.64M parameters. We compare the model size with state-of-the-art methods in Table 4.2.

Model	# parameters	Storage (MB)
DBN [97]	15.3M	58.4
RDN [112]	16.4M	62.6
OVD [46]	0.90M	3.4
IFI-RNN (single cell)	0.84M	3.2
IFI-RNN (dual cell)	1.64M	6.2

Table 4.2: Model size comparison with other deep learning based methods. For RDN, we refer to the model and source code provided by the authors of [112], which is different from the paper. In the main paper and this supplementary material, the comparisons are consistently done with the provided model.

We provide two different types of iteration: the single cell and the dual cell method. In the single cell method, we use the same parameters to estimate both the initial hidden state and the updated hidden state. On the other hand, in a dual cell method, we use two RNN cells and use each of them for a different purpose. Only the second cell is used to update the hidden states and predict latent frames. Although the dual cell method requires more parameters than the single cell approach, it can bring significant performance gain as different sets of parameters can dedicate to different roles. From now on, we denote the single and dual cell models with prefix C1 and C2, respectively. Also, we put a suffix H with the hidden state iterations. For example, C2H2 denotes the dual cell model which updates its hidden state two times.

We describe the two intra-frame hidden state updating methods in Algorithm 1. In an architectural viewpoint, our methods virtually increase the depth of RNN cell, enlarging the receptive field and its capacity. In other words, our hidden states can be better optimized by a virtually deeper model.

Algorithm 1 Deblurring with intra-frame hidden state update

procedure SINGLE CELL METHOD($\mathbf{B}_t, \mathbf{h}_{t-1}$)
  $f_{\mathbf{B}_t} = \mathcal{F}_{\mathbf{B}}\left(\mathbf{B}_t\right)$
  $\hat{\mathbf{h}}_t^0 \leftarrow \mathbf{h}_{t-1}$
  for $i = 1 \ldots N$ **do**
    $f_R^i = \mathcal{F}_R\left(f_{\mathbf{B}_t}, \hat{\mathbf{h}}_t^{i-1}\right)$
    $\hat{\mathbf{h}}_t^i = \mathcal{F}_{\mathbf{h}}\left(f_R\right)$
  $\mathbf{h}_t \leftarrow \hat{\mathbf{h}}_t^N$
  $\mathbf{L}_t = \mathcal{F}_L\left(f_R^N\right)$
  return $\mathbf{L}_t, \mathbf{h}_t$

procedure DUAL CELL METHOD($\mathbf{B}_t, \mathbf{h}_{t-1}$)
  $f_{\mathbf{B}_t,1} = \mathcal{F}_{\mathbf{B},1}\left(\mathbf{B}_t\right)$
  $f_{\mathbf{B}_t,2} = \mathcal{F}_{\mathbf{B},2}\left(\mathbf{B}_t\right)$
  $\hat{\mathbf{h}}_t^0 = \mathcal{F}_{\mathbf{h},1}\left(\mathcal{F}_{\mathbf{R},1}\left(f_{\mathbf{B}_t}, \mathbf{h}_{t-1}\right)\right)$
  for $i = 1 \ldots N$ **do**
    $f_{R,2}^i = \mathcal{F}_{\mathbf{R},2}\left(f_{\mathbf{B}_t,2}, \hat{\mathbf{h}}_t^{i-1}\right)$
    $\hat{\mathbf{h}}_t^i = \mathcal{F}_{\mathbf{h},2}\left(f_{R,2}^i\right)$
  $\mathbf{h}_t \leftarrow \hat{\mathbf{h}}_t^N$
  $\mathbf{L}_t = \mathcal{F}_{\mathbf{L}}\left(f_{R,2}^N\right)$
  return $\mathbf{L}_t, \mathbf{h}_t$

4.3.3 Regularization by Stochastic Training

The performance gains from iteration, however, become marginal for higher iteration models. For example, the C1H4 model (single cell four iterations) does not perform better than C1H3 model in Figure 4.5. We also observe that for each image, the best performing model is not always the one with more iterations. Figure 4.4 shows the number of images that are best restored by the single-cell method with different iterations. Although many images prefer more iterations for better restoration, a nontrivial amount of images favor lesser iterations. Since we use the same RNN cell for each iteration, it is natural to conjecture that we can train a model that can deblur each input frame with different iterations in a stochastic way. Therefore, we attempt to take advantage of the regularization effect from using stochastic computational path for training.

First, we add a gating unit $g(\cdot) \in \{0, 1\}$ that looks into the hidden state and decides if the model will compute one more iteration or not. We calculate the score for iteration by global average pooling [62] followed by two fully connected layers activated by ReLU [51]. Then discrete binary sampling is done with the Gumbel-SoftMax trick [39, 65]. At the training time, when the gate is on, we update the hidden state once more. Otherwise, we stop the iteration and return the deblurred frame. Second, we employ a regularization term, that favors fewer iterations when the loss is already small enough. We set a target average iteration ratio, $\tau = 0.75$. Compared to the models with a fixed iteration number, this loss prefers stopping the iteration with the probability of $1 - \tau$. We define the term as L2 loss between the average gate activation over a mini-batch with iterations and τ,

$$\mathcal{L}_{\text{reg}} = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} \left(\mathbb{E}\left[g_t^i \right] - \tau \right)^2,$$

where $\mathbb{E}\left[\cdot\right]$ is an average operation, $g_t^i = g\left(\hat{\mathbf{h}}_t^i\right)$ at iteration i at time-step t, and N is the maximum iteration threshold we set during training.

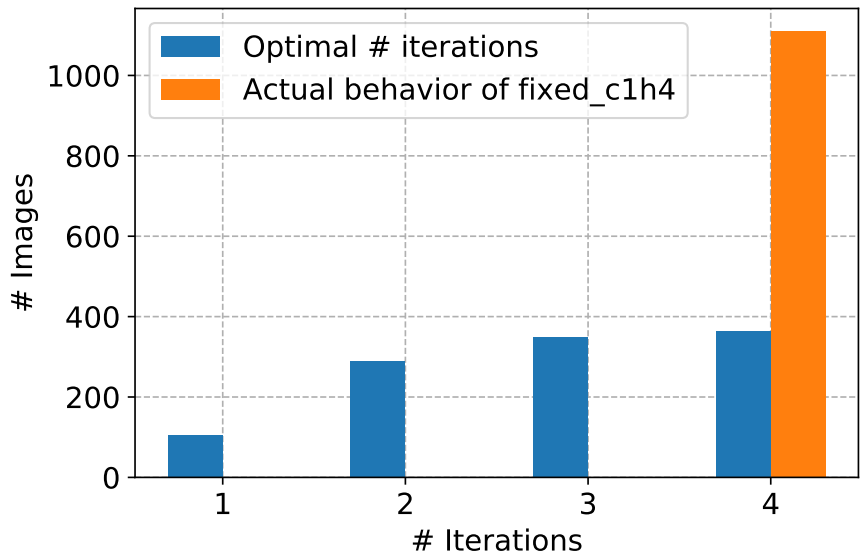


Figure 4.4: **Histogram showing the number of best restored images by the number of iterations.** Blue bars show the number of images that are best restored by the single-cell method according to the iterations. Orange bar represents the total number of images restored by C1H4 model. We used downsampled GOPRO test images [72]. Refer to section 4.4.1 for details.

Thus, our final loss term becomes

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{content}} + \lambda \times \mathcal{L}_{\text{reg}},$$

with λ being the weight for the regularization term. Note that our primary purpose of the stochastic training is to improve the results by regularizing the co-adaptation of parameters, rather than making our model to show stochastic behavior. So, we remove the gating unit after training so that the system provides the results of a specified number of iterations.

The performances of regularized models are shown as dotted lines in Figure 4.5. We add '-reg' suffix to our IFI-RNN models to refer models trained with regularization like C2H3-reg.

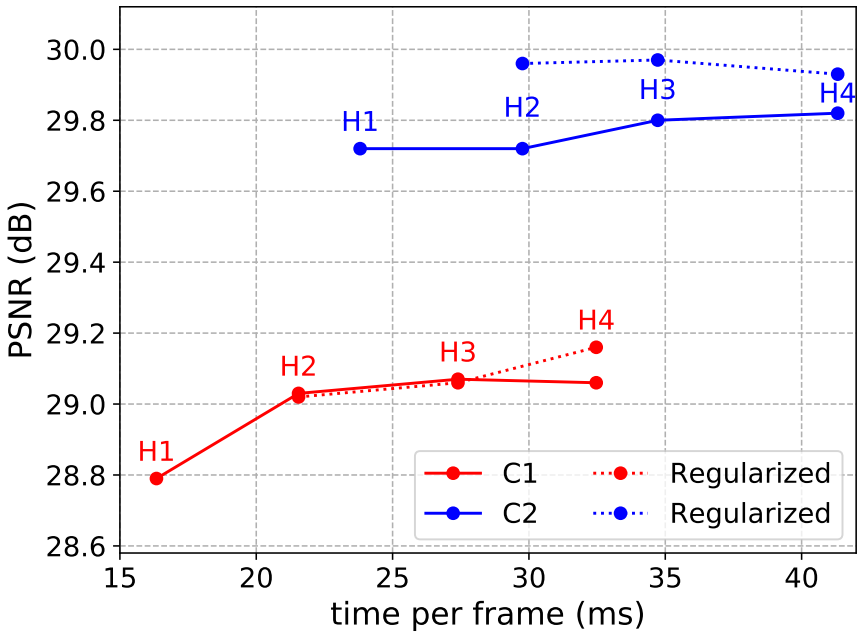


Figure 4.5: **PSNR and the running time of our methods, evaluated on downsampled GOPRO test set at resolution 960×540.** Refer to section 4.4.1 for details.

4.4 Experiments

4.4.1 Datasets

We have tested our algorithm (denoted as IFI-RNN) on the GOPRO dataset [72]. The GOPRO dataset contains 2103 training samples from 22 sequences and 1111 evaluation samples from 11 sequences. We generated blur and sharp image pairs from 240 fps videos. Those high-speed video frames are averaged in a gamma-transformed domain to mimic images taken in longer exposure time with nonlinear camera response function (CRF). To suppress the noise and video compression artifacts, we downsampled the original video resolution from 1280×720 to 960×540 before averaging.

We also use a similar dataset from Su et al. [97]. This dataset also consists of paired samples synthesized from 240 fps videos. It provides 61 sequences containing 5708 training pairs and 10 sequences including 1000 evaluation pairs. However, we do not

evaluate with the method proposed by Köhler et al. [50] as [97]. Instead, we evaluate PSNR and SSIM as is without post-processing such as alignment. In addition to the original captured frames, they interpolate intermediate sharp frames from optical flow estimation to generate smooth blur frames. The original and interpolated frames are averaged altogether to synthesize blurs under linear CRF assumption.

To compare our method with previous methods, we use the test video sequences from the above two datasets except for the first four frames and the last frame in each video, as [46] does not provide the results for them. Also, we show the deblurring results of real videos to demonstrate the generalization capability of our method.

4.4.2 Implementation details

We train our models on the GOPRO dataset [72] with ADAM optimizer [49] where $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We train each model for 500 epochs in total. Beginning from the initial learning rate of 10^{-4}, we anneal the learning rate by half after every 200 epochs. We set the regularization loss weight $\lambda = 10$. During training, we sample 12-frame 256×256 RGB patch sequences from the dataset to construct a mini-batch of size 4. Random augmentations are applied to those samples with geometric transforms including vertical and horizontal flips as well as 90^o rotation. Also, we add zero-mean Gaussian noise to blurry inputs, where its standard deviation is sampled from another Gaussian distribution $\mathcal{N}\left(0, 2^2\right)$ to blurry inputs NVIDIA GTX 1080 Ti GPUs were used for all of our experiments. We implemented our models with PyTorch 0.4.1 [87] built with CUDA 9.2 and cuDNN 7.1. Our source code will be released publicly.

4.4.3 Comparisons on GOPRO [72] dataset

We evaluate our method and other methods on the downsampled GORPO dataset. We report the evaluation results of all the comparing methods in terms of PSNR, SSIM and the running time in Table 4.3. From these results, it is clear that the proposed intra-frame iteration scheme and the stochastic training method improve the perfor-

mance of our model significantly compared with the other state-of-the-art methods. Furthermore, surprisingly, our method is much faster than the others, despite having internal iterative operations. For visual comparison, please refer to Figure 4.1.

Method	PSNR	SSIM	Speed (fps)
DBN+OF [97]	27.08	0.8429	1.72†
RDN [112]	25.19	0.7794	7.37
OVD [46]	26.82	0.8245	9.24
IFI-RNN (C1H1)	28.79	0.8647	61.2
IFI-RNN (C1H2)	29.03	0.8712	46.4
IFI-RNN (C1H3)	29.07	0.8730	36.5
IFI-RNN (C1H4)	29.06	0.8730	30.8
IFI-RNN (C1H4-reg)	**29.16**	**0.8760**	30.8
IFI-RNN (C2H1)	29.72	0.8884	42.0
IFI-RNN (C2H2)	29.72	0.8885	33.6
IFI-RNN (C2H3)	29.80	0.8900	28.8
IFI-RNN (C2H4)	29.82	0.8913	24.2
IFI-RNN (C2H3-reg)	**29.97**	**0.8947**	28.8
IFI-RNN (C2H4-reg)	29.93	0.8943	24.2

Table 4.3: **Deblurring accuracy comparison on the downsampled GOPRO dataset [72].** For our method IFI-RNN, C1 and C2 refer to single-cell and dual-cell method, respectively. †Note that the above speed does not include the optical flow estimation time for [97]. All running times were averaged from 10 runs on the test set.

4.4.4 Comparisons on [97] Dataset and Real Videos

We also compared the performances on the dataset presented in [97]. In this case, we fine-tuned our pretrained models from the GOPRO dataset with the training subset of [97]. In Table 4.4, our model also improves performance with iterations and regularization for both C1 and C2 models. Furthermore, IFI-RNN C2 models show state-of-the-art performance. In Figure 4.6, our IFI-RNN recovers the text and legs more clearly from the blurry video frame. Also, our results on real videos also clarify blurred textures in Figure 4.7. In many examples, our IFI-RNN shows sharper recon-

struction results especially on the fine textured area. We notice better reconstructed faces in Fig. 4.8, 4.9, 4.11, 4.12. Also, our method shows lesser artifact in recovering more easily readable text in Fig. 4.8, 4.10, 4.11. Fine-grained textures are more recognizable in Fig. 4.9, 4.13.

Method	PSNR	SSIM
DBN+OF [97]	30.14	0.8913
RDN [112]	26.98	0.8076
OVD [46]	29.97	0.8696
IFI-RNN (C1H1)	30.07	0.8823
IFI-RNN (C1H4-reg)	30.10	0.8849
IFI-RNN (C2H1)	30.74	0.8974
IFI-RNN (C2H3-reg)	**30.80**	**0.8991**
IFI-RNN (C2H4-reg)	30.73	0.8976

Table 4.4: **Deblurring accuracy comparison on the dataset from [97].**

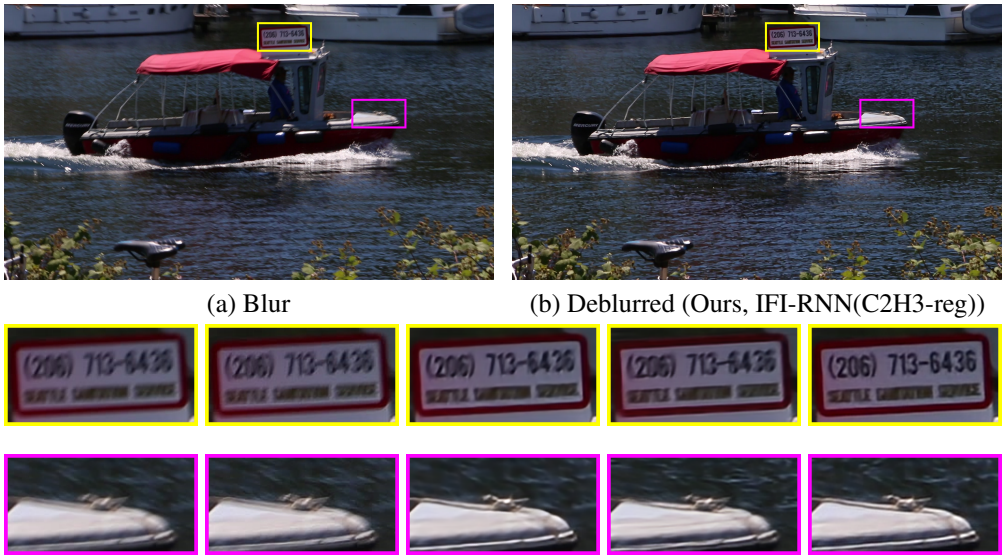
4.5 Conclusion

In this chapter, we proposed a method to ameliorate the recurrent network architectures for video deblurring. By iteratively updating the hidden state to adapt to the target frame, our method removes blurs in the video frames more effectively. Furthermore, we train our model with a regularization term that could enhance prediction accuracy through stochastic computation paths. Our method does not require additional parameters while being fast and accurate compared to other state-of-the-art methods.

(a) Blur (b) Deblurred (Ours, IFI-RNN(C2H3-reg))

(c) Blur (d) RDN [112] (e) OVD [46] (f) DBN+OF [97] (g) IFI-RNN

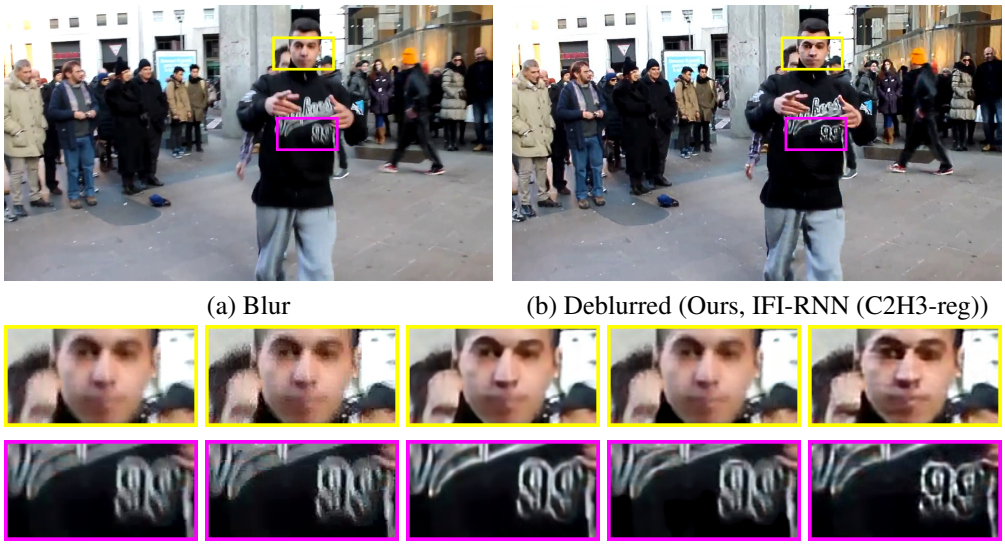
Figure 4.6: **Deblurred results on [97] dataset.**



(a) Blur (b) Deblurred (Ours, IFI-RNN(C2H3-reg))

(c) Blur (d) RDN [112] (e) OVD [46] (f) DBN+OF [97] (g) IFI-RNN

Figure 4.7: **Deblurred results of a real video.**



(a) Blur (b) Deblurred (Ours, IFI-RNN (C2H3-reg))

(c) Blur (d) RDN [112] (e) OVD [46] (f) DBN+OF [97] (g) IFI-RNN

Figure 4.8: Deblurring results of a real video.

(a) Blur (b) Deblurred (Ours, IFI-RNN (C2H3-reg))

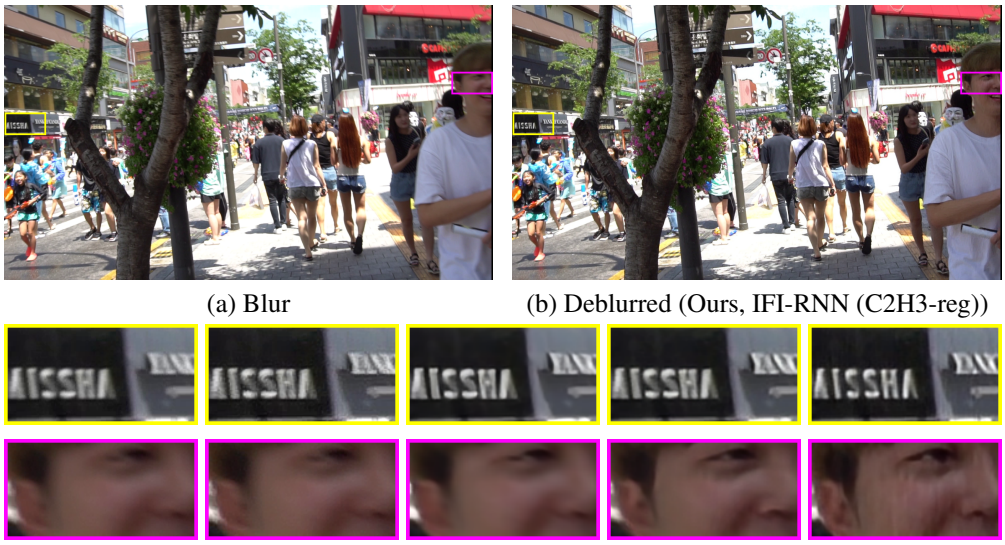
(c) Blur (d) RDN [112] (e) OVD [46] (f) DBN+OF [97] (g) IFI-RNN

Figure 4.9: Deblurring results of a real video.

(a) Blur (b) Deblurred (Ours, IFI-RNN (C2H3-reg))

(c) Blur (d) RDN [112] (e) OVD [46] (f) DBN+OF [97] (g) IFI-RNN

Figure 4.10: Deblurring results of a real video.



(a) Blur (b) Deblurred (Ours, IFI-RNN (C2H3-reg))

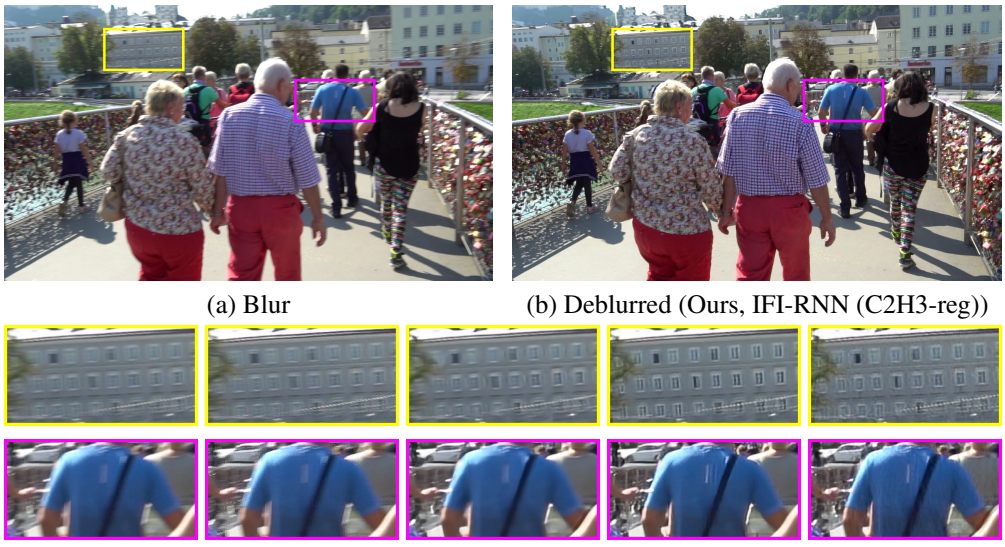
(c) Blur (d) RDN [112] (e) OVD [46] (f) DBN+OF [97] (g) IFI-RNN

Figure 4.11: Deblurring results of a real video.

(a) Blur (b) Deblurred (Ours, IFI-RNN (C2H3-reg))

(c) Blur (d) RDN [112] (e) OVD [46] (f) DBN+OF [97] (g) IFI-RNN

Figure 4.12: Deblurring results of a real video.



(a) Blur (b) Deblurred (Ours, IFI-RNN (C2H3-reg))

(c) Blur (d) RDN [112] (e) OVD [46] (f) DBN+OF [97] (g) IFI-RNN

Figure 4.13: Deblurring results of a real video.

Chapter 5

Learning Loss Functions for Image Deblurring

5.1 Introduction

In classical energy minimization-based methods for image deblurring, prior terms were designed as well as the likelihood terms as (2.1). The prior terms have played key roles in mitigating the ill-posedness of the problem and to encode the preferences on sharp and clear images by using statistical properties [37, 111, 43, 44]. In recent deep learning-based deblurring methods, on the other hand, priors are not widely used to train neural networks. Instead of designing loss terms with the human knowledge-driven priors, most of the learning process chooses to rely on supervision from large-scale datasets by minimizing the distance between the output and the ground truth in Euclidean space, e.g., L1 or L2, to maximize PSNR between the deblurred and the reference sharp images.

With the advent of modern CNN architectures, state-of-the-art deblurring methods focused on designing sophisticated neural network architectures [72, 103, 28, 117, 86] for better model capacity and deblurring accuracy. Although learning from temporally center-aligned blur-sharp image pairs can relieve the ill-posedness of deblurring, it is not fully solved, yet. Still, most methods tend to suffer from blurry predictions due to the inherent limitation [56, 68] of PSNR-oriented solutions for ill-posed problems.

(a) Sharp / Blur (b) SRN (c) DeblurGANv2 (d) **Ours**

Figure 5.1: **Comparison of the deblurred images and their reblurred counterparts.** For each image, we visualize the remaining blur kernel [13] at the center pixel visualized on the right bottom side. **Upper**: The kernels from the previous methods implicate the direction of the original blur. **Lower**: When the proposed reblurring module is applied, our result does not lose sharpness as we reconstruct the output that is hard to be reblurred.

To complement the conventional learning objectives, several attempts such as the perceptual [42] and the adversarial loss [56, 72, 52] have been made to improve the visual quality and the sharpness of the model output. Nevertheless, the previous perceptual losses may not be optimal for blur removal as the low-level structural properties such as blurriness are not explicitly considered in their formulations. Rather, they originate from features learned for high-level tasks such as image classification and real/fake image discrimination. As illustrated in Figure 5.1, results from the existing deblurring methods are not as sharp as the ground-truth example but are still blurry to a degree. Despite the reduced strength of blur in the deblurred images, we still observe the directional motion information remaining.

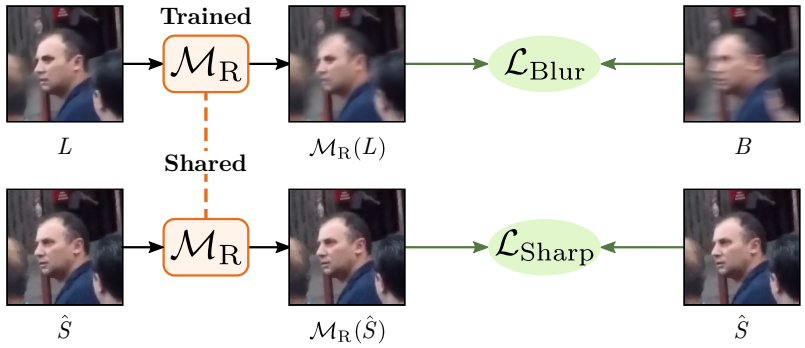
The observation tells that applying the VGG and the adversarial loss together [53] is not sufficient to obtain perceptually pleasing and sharp images across different architectures [104, 53]. By finding the inherent limitation of the previous loss terms, we

conjecture that eliminating the motion cues remaining in the deblurred images could play an essential role in generating sharp and clean images. Starting from the motivation, we introduce the concept of *reblurring* which amplifies the unremoved blur in the given image. An ideally deblurred image should be sharp enough so that no noticeable blur can be found from it to be amplified, i.e., clean images are hard to reblur. In contrast, it is easier to predict the original shape of blur by recognizing the remaining blur kernel if the motion blur is not sufficiently removed. In this chapter, we propose to use the difference as a new optimization objective, designing *reblurring loss* for image deblurring problem.

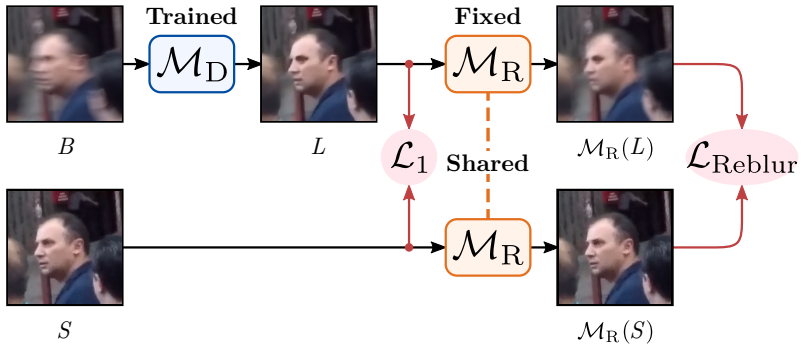
The reblurring loss is realized by jointly training a deblurring module and the paired reblurring module. From a deblurred output, the reblurring module tries to make the reblurred image as close to the original blurry image. By using the property that the reblurred results should vary by the degree of input blur to be amplified, we construct two types of loss functions.

During the joint training, *supervised reblurring loss* compares the amplified blurs between the deblurred and the sharp image. Complementing the L1 intensity loss, the supervised reblurring loss guides the deblurring module to focus on and eliminate the remaining blur information. While the training method being similar to the adversarial training of GANs [30], the purposes and effects of the adversary are different. Our reblurring loss concentrates on the image blurriness regardless of the image realism in the training process. Furthermore, we apply *self-supervised reblurring loss* at test-time so that the deblurred image would be infeasible to be reblurred as sufficiently sharp image would. The self-supervised reblurring loss lets the deblurring module to adaptively optimize to each input without ground truth.

The reblurring loss functions provide additional optimization directives to the deblurring module and can be generally applied to any learning-based methods. With the proposed approach, sharper images can be obtained without modifying the structure of the deblurring module.



(a) Reblurring module training process



(b) Deblurring module training with reblurring loss

Figure 5.2: **Overview of the proposed reblurring and deblurring framework.**

We summarize our contributions in this chapter as follows:

- Based on the observation that clean images are hard to reblur, we propose novel loss functions for image deblurring. Our reblurring loss reflects the preference for sharper images and contributes to visually pleasing deblurring results.
- At test-time, the reblurring loss can be implemented without a ground-truth image. We perform test-time adaptive inference via self-supervised optimization to each input.
- Our method is generally applicable to any learning-based methods and jointly with other loss terms. Experiments show that the concept of reblurring loss consistently contributes to achieving state-of-the-art visual sharpness as well as LPIPS and NIQE across different model architectures.

5.2 Related Works

Image Deblurring. In the classic energy optimization framework, the energy is formulated by the likelihood and the prior term. Due to the large solution space of the ill-posed dynamic scene deblurring problem, prior terms have been the essential element in alleviating the optimization ambiguity by encoding the preference on the solutions. Sophisticated prior terms were carefully designed with human knowledge on natural image statistics [58, 13, 37, 111, 100, 116, 43, 44, 83]. In the recent work of Li *et al.* [59], learned prior, which is derived from a classifier discriminating blurry and clean images, was also shown to be effective. Deep priors were also used for image deconvolution problems [89, 79].

On the other hand, deep learning methods have benefited from learning on large-scale datasets. The datasets consisting of realistic blur [72, 97, 82, 71, 28, 41, 94] align the temporal center of the blurry and the sharp image pairs with high-speed cameras. Learning from such temporally aligned datasets relieves the ill-posedness of deblurring compared with the large solution space in the energy optimization framework. Thus, more attention has been paid to designing CNN architectures and datasets than the loss or solution preference.

In the early work, the alternating kernel and image estimation processes [13] are implemented with CNNs [92]. In [99, 29], the spatially varying blur kernels are estimated by assuming locally linear blur followed by non-blind deconvolution with them. Later, end-to-end learning without explicit kernel estimation became prevalent. Motivated from the coarse-to-fine approach, multi-scale CNN was proposed in [72] to expand the receptive field efficiently. Several studies have proposed scale-recurrent architectures [104, 28] that share parameters across the scales. On the other hand, [119, 98] sequentially stacked network modules. Recently, [86] proposed a multi-temporal model that deblurs an image recursively. To handle spatially varying blur kernels efficiently, spatially non-uniform operations were embedded [123, 117] in the neural networks.

Perceptual Image Restoration. Conventional image restoration methods mainly optimize L1 or L2 objectives to achieve higher PSNR. However, such approaches suffer from blurry and over-smoothed outputs [42, 126, 68]. The primary reason is that the learned models predict an average of all possible solutions under the ill-posedness [56]. To deal with the issue, several studies utilize deep features of the pretrained VGG [95] network that are more related to human perception [42, 56, 126]. Then, the following methods can produce perceptually better results by minimizing the distance of output and ground-truth images in the feature domain. Recent methods further introduce adversarial training [30] so that outputs of the restoration models be indistinguishable from real samples [72, 81, 52, 53].

Nevertheless, an inherent limitation of existing perceptual objectives is that they are not task-specialized for image restoration. For example, the VGG features [95] are learned for high-level visual recognition [91] while adversarial loss [30] only contributes to reconstruct realistic images without considering the existence of motion blur. Therefore, blindly optimizing those terms may not yield an optimal solution in terms of image deblurring. In practice, we observed that those objectives still tend to leave blur footprints unremoved, making it possible to estimate the original blur. Our reblurring loss is explicitly designed to improve the perceptual quality of deblurred images by reducing remaining blurriness and thus more suitable for the deblurring task.

Image Blurring. As an image could be blurred in various directions and strength, image blurring is another ill-posed problem. Thus intrinsic [3] or extrinsic [11, 8] information is often incorporated. In the case of a non-ideally sharp image, Bae *et al.* [3] detected the small local blur kernel in the image to magnify the defocus blur for the bokeh effect. On the other hand, [11] estimated the kernel by computing the optical flow from the neighboring video frames. In a similar sense, [8] used multiple video frames to synthesize blur. Without such blur or motion cue, there could be infinitely many types of plausible blur applicable to an image. Thus, [124] used a generative

#ResBlocks	4	8	16	32
Deblur PSNR wrt sharp GT	28.17	29.67	30.78	31.48
Reblur PSNR wrt blur GT	34.29	32.66	31.90	31.48

Table 5.1: **Deblurring and reblurring PSNR (dB) by deblurring model capacity.** Both tasks are trained independently with L1 loss on the GOPRO [72] dataset. We note that #ResBlocks varies for the deblur network only.

model to synthesize many realistic blurry images. Contrary to the above approaches, [4] deliberately blurred an already blurry image in many ways to find the local blur kernel. Our image reblurring concept is similar to [3] in the sense that intrinsic cue in an image is used to amplify blur. Nonetheless, our main goal is to use reblurring to provide a guide to deblurring model so that such blur cues would be removed.

5.3 Proposed Method

In this section, we describe a detailed concept of imge reblurring and how the reblurring operation can be learned. The proposed reblurring loss can support the deblurring modules to reconstruct perceptually favorable and sharp outputs. At training and testing stages, we formulate the reblurring loss in supervised and self-supervised manner. For simplicity, we refer to the blurry, the deblurred, and the sharp image as B, L, and S, respectively.

5.3.1 Clean Images are Hard to Reblur

As shown in Figure 5.1, outputs from the existing deblurring methods still contain undesired motion trajectories that are not completely removed from the input. Ideally, a well-deblurred image should not contain any motion cues making reblurring to be infeasible. To validate our motivation that clean images are hard to reblur, we first build a reblurring module $\mathcal{M}_{\text{R}}$ which amplifies the remaining blur from L. The module is trained with the following blur reconstruction loss $\mathcal{L}_{\text{Blur}}$ so that it would learn the

inverse operation of deblurring as

$$\mathcal{L}_{\text{Blur}} = \|\mathcal{M}_{\text{R}}(L) - B\|. \tag{5.1}$$

We apply $\mathcal{M}_{\text{R}}$ to the deblurred images from deblurring modules of varying capacities. Table 5.1 shows that the higher the deblurring PSNR, the lower the reblurring PSNR becomes. It demonstrates the better deblurred images are harder to reblur, justifying our motivation.

In contrast to the non-ideally deblurred images, $\mathcal{M}_{\text{R}}$ is not able to generate a motion blur from a sharp image S. For a high-quality clean image, $\mathcal{M}_{\text{R}}$ should preserve the sharpness. However, optimizing the blur reconstruction loss $\mathcal{L}_{Blur}$ with S may fall into learning the pixel average of all blur trajectories in the training dataset. In such a case, $\mathcal{M}_R$ will apply a radial blur without considering the input variety. To let the blur domain of $\mathcal{M}_R$ be confined to the motion-incurred blur, we use sharp images to penalize such undesired operation. Specifically, we introduce a network-generated sharp image $\hat{S}$ obtained by feeding a real sharp image S to the deblurring module $\mathcal{M}_{\text{D}}$, as $\hat{S} = \mathcal{M}_{\text{D}}(S)$. We define sharpness preservation loss $\mathcal{L}_{\text{Sharp}}$ as follows:

$$\mathcal{L}_{\text{Sharp}} = \|\mathcal{M}_{\text{R}}(\hat{S}) - \hat{S}\|. \tag{5.2}$$

We use pseudo-sharp image $\hat{S}$ instead of a real image S to make our reblurring module focus on image sharpness and blurriness rather than image realism. While $\hat{S}$ and L only differ by the sharpness, S and L also vary by their realism.

Combining two loss terms together, we train the reblurring module $\mathcal{M}_{\text{R}}$ by optimizing the joint loss $\mathcal{L}_{\text{R}}$:

$$\mathcal{L}_{\text{R}} = \mathcal{L}_{\text{Blur}} + \mathcal{L}_{\text{Sharp}}. \tag{5.3}$$

As zero-magnitude blur should remain unaltered from $\mathcal{M}_{\text{R}}$, the sharpness preservation loss can be considered a special case of the blur reconstruction loss. Figure 5.2a

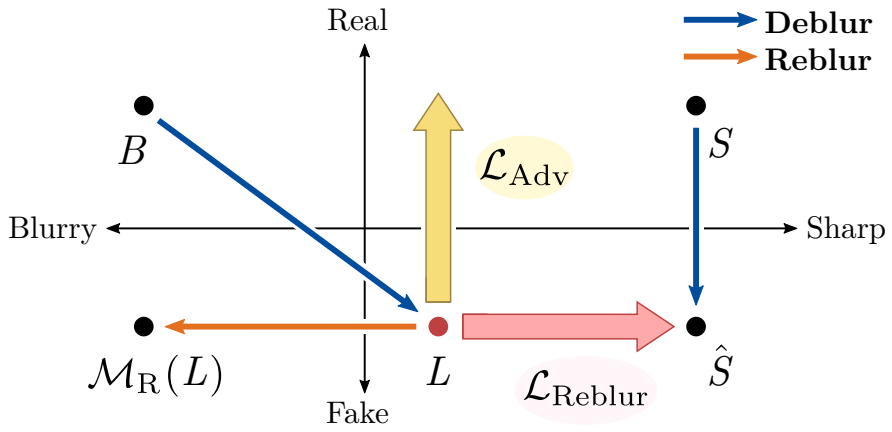


Figure 5.3: **Image deblurring and reblurring illustrated from the perspective of sharpness and realism.** Training our modules with $\mathcal{L}_{\text{Reblur}}$ improves image sharpness without considering the image realism. The image realism can be optionally handled by adversarial loss $\mathcal{L}_{\text{Adv}}$.

illustrates the way our reblurring module is trained from $\mathcal{L}_{\text{R}}$.

5.3.2 Supervision from Reblurring Loss

The blurriness of images can be easily compared by amplifying the blur. Thus, we propose a new optimization objective by processing the deblurred and the sharp image with the proposed reblurring model $\mathcal{M}_{\text{R}}$. To suppress remaining blur in the output L of the deblurring CNN $\mathcal{M}_{\text{D}}$, our reblurring loss $\mathcal{L}_{\text{Reblur}}$ for image deblurring is defined as follows:

$$\mathcal{L}_{\text{Reblur}} = \|\mathcal{M}_{\text{R}}(L) - \mathcal{M}_{\text{R}}(S)\|. \tag{5.4}$$

Unlike the sharpness preservation term in (5.2), we do not use the pseudo-sharp image $\hat{S}$ in our reblurring loss (5.4). As the quality of the pseudo-sharp image $\hat{S}$ depends on the state of deblurring module $\mathcal{M}_{\text{D}}$, using $\hat{S}$ may make training unstable and difficult to optimize, especially at the early stage. Thus we use a real sharp image S to stabilize the training. Nevertheless, as $\mathcal{M}_{\text{R}}$ is trained to focus on the sharpness from (5.3), so does the reblurring loss $\mathcal{L}_{\text{Reblur}}$.

Using our reblurring loss in (5.4), the deblurring module $\mathcal{M}_{\text{D}}$ is trained to mini-

mize the following objective $\mathcal{L}_{\mathrm{D}}$:

$$\mathcal{L}_{\mathrm{D}} = \mathcal{L}_1 + \lambda \times \mathcal{L}_{\mathrm{Reblur}}, \tag{5.5}$$

where $\mathcal{L}_1$ is a conventional L1 loss, and the hyperparameter λ is empirically set to 1. Figure 5.2b shows how the deblurring model is trained with our proposed reblurring loss.

For each training iterations, we alternately optimize two modules $\mathcal{M}_{\mathrm{D}}$ and $\mathcal{M}_{\mathrm{R}}$ by $\mathcal{L}_{\mathrm{D}}$ and $\mathcal{L}_{\mathrm{R}}$, respectively. While such a strategy may look similar to the adversarial training [30], the optimization objectives are different. As the neural networks are well known to easily discriminate real and fake images [109], the realism could serve as a more salient feature than image blurriness. Thus, adversarial loss may overlook image blurriness as L and S can already discriminated by realism difference. On the other hand, our reblurring loss is explicitly designed to prefer sharp images regardless of realism. Figure 5.3 conceptually compares the actual role of the proposed reblurring loss $\mathcal{L}_{\mathrm{Reblur}}$ and the existing adversarial loss $\mathcal{L}_{\mathrm{Adv}}$.

5.3.3 Test-time Adaptation by Self-Supervision

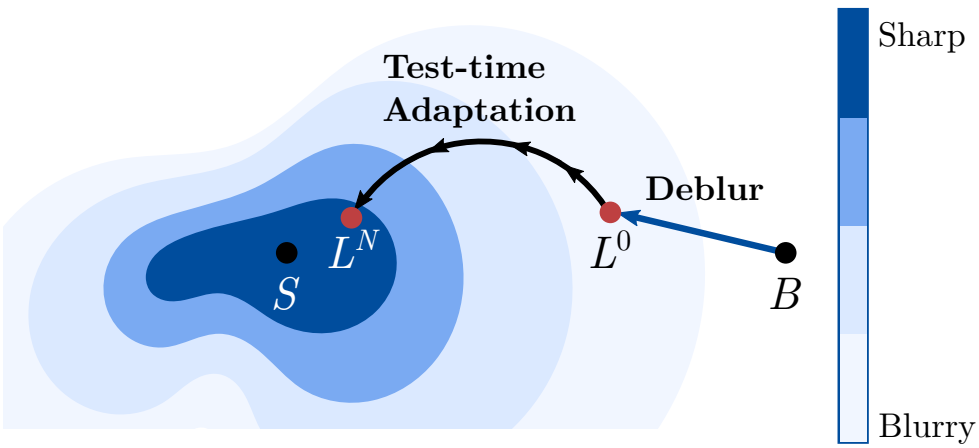


Figure 5.4: **The proposed self-supervised test-time adaptation.** We repetitively find the latent image that reblurs to the current deblurred image.

Algorithm 2 Optimization process in test-time adaptation

procedure TEST-TIME ADAPTATION($B, \mathcal{M}_{\mathrm{D}}, \mathcal{M}_{\mathrm{R}}$)
  Test-time learning rate $\mu \leftarrow 3 \times 10^{-6}$.
  $\theta_{\mathrm{D}} \leftarrow$ Weights of $\mathcal{M}_{\mathrm{D}}$.
  $L^0 = \mathcal{M}_{\mathrm{D}}(B)$.
  for $i = 0 \ldots N-1$ **do**
    $L_*^i = \mathcal{M}_{\mathrm{D}}(B)$.
    $\mathcal{L}_{\text{reblur}}^{\text{self}} = \|\mathcal{M}_{\mathrm{R}}(\mathcal{M}_{\mathrm{D}}(B)) - L_*^i\|$.
    Update θ_{D} by $\nabla_{\theta_{\mathrm{D}}}\mathcal{L}_{\text{Reblur}}^{\text{self}}$ and μ.
  $L^N = \mathcal{M}_{\mathrm{D}}(B)$.
  $L_{\text{Adapted}}^N =$`histogram_matching`(L^N, L_*^0)
  return L_{Adapted}^N

Supervised learning methods have the fixed model weights at testing time as the training with ground truth is no longer available. Every image is treated equally regardless of the scene content and the blur difficulty at test time. In contrast, providing self-supervised loss can make a model adapt to each test input, improving the generalization ability. Thus, we use the proposed reblurring operation to enable a novel self-supervised optimization without the need for ground truth.

During the training of $\mathcal{M}_{\mathrm{D}}$, $\mathcal{M}_{\mathrm{R}}$ delivers supervision from the reference by (5.4). With the learned reblurring operation, we can further exploit the deblurred image L is in high-quality in terms of shaprness without reference data. If L gets blurred by passing to $\mathcal{M}_{\mathrm{R}}$, we can consider it to be insufficiently deblurred as we have discussed in Figure 5.1. A clean image should remain as itself due to the sharpness preservation loss, $\mathcal{L}_{Sharp}$. Thus, we construct the self-supervised reblurring loss that could serve as a prior term encoding the preference on sharp images.

$$\mathcal{L}_{\text{Reblur}}^{\text{self}} = \|\mathcal{M}_{\mathrm{R}}(L) - L_*\|, \tag{5.6}$$

where L_* denotes the image with the same value as L but the gradient does not back-propagate in the optimization process. We minimize $\mathcal{L}_{\text{Reblur}}^{\text{self}}$ for each test data to obtain the sharper image. Allowing gradient to flow through L_* can let L to fall into undesired

local minima where both the L and $\mathcal{M}_{\mathrm{R}}(L)$ are blurry. Since $\mathcal{L}^{\mathrm{self}}_{\mathrm{Reblur}}$ only considers the sharpness of an image, we keep the color consistency by matching the color histogram between the test-time adapted image and the initially deblurred image. The detailed process of test-time adaptation strategy is described in Algorithm 2 and conceptually illustrated in Figure 5.4.

5.4 Experiments

We demonstrate the effectiveness of our reblurring loss by applying it to multiple model architectures. We show the experimental results with a baseline residual U-Net and the state-of-the-art image deblurring models, the sRGB version of SRN [104] and DHN, our modified version of DMPHN [119]. For the reblurring module, we use simple residual networks with 1 or 2 ResBlock(s) with 5×5 convolution kernels. The training and evaluation were done with the widely used GOPRO [72] and REDS [71] datasets. The GOPRO dataset consists of 2103 training and 1111 test images with various dynamic motion blur. Similarly, the REDS dataset has 24000 training and 3000 validation data publicly available. On each dataset, every experiment was done under the same training environment. We mainly compare LPIPS [126] and NIQE [70] perceptual metrics.

5.4.1 Effect of Reblurring Loss

We implement the reblurring loss in varying degrees of emphasis on sharpness by controlling the reblurring module capacity. For a more balanced quality between PSNR and perceptual sharpness, we use 1 ResBlock for $\mathcal{M}_{\mathrm{R}}$. To put more weight on the perceptual quality, we allocate a larger capacity on $\mathcal{M}_{\mathrm{R}}$ by using 2 ResBlocks. For notation simplicity, we denote the reblurring loss with k ResBlock(s) in the reblurring module as $\mathcal{L}_{\mathrm{Reblur,\ n}k}$.

Table 5.2 and 5.3 each shows how the deblurring performance varies depending on

Method	LPIPS$_\downarrow$	NIQE$_\downarrow$	PSNR$^\uparrow$	SSIM$^\uparrow$
U-Net ($\mathcal{L}_1$ only)	0.1635	5.996	29.66	0.8874
$+\mathcal{L}_{\text{Reblur, n1}}$	0.1365	5.629	29.58	0.8869
$+\mathcal{L}_{\text{Reblur, n2}}$	**0.1238**	**5.124**	29.44	0.8824
SRN ($\mathcal{L}_1$ only)	0.1246	5.252	30.62	0.9078
$+\mathcal{L}_{\text{Reblur, n1}}$	0.1140	5.136	30.74	0.9104
$+\mathcal{L}_{\text{Reblur, n2}}$	**0.1037**	**4.887**	30.57	0.9074
DHN ($\mathcal{L}_1$ only)	0.1179	5.490	31.53	0.9207
$+\mathcal{L}_{\text{Reblur, n1}}$	0.0975	5.472	31.53	0.9217
$+\mathcal{L}_{\text{Reblur, n2}}$	**0.0837**	**5.076**	31.34	0.9177

Table 5.2: **Perceptual metric improvements from the reblurring loss on GOPRO [72] dataset.** The reblurring loss consistently improves LPIPS and NIQE over standard $\mathcal{L}_1$ loss.

Method	LPIPS$_\downarrow$	NIQE$_\downarrow$	PSNR$^\uparrow$	SSIM$^\uparrow$
U-Net ($\mathcal{L}_1$ only)	0.1486	3.649	30.80	0.8772
$+\mathcal{L}_{\text{Reblur, n1}}$	0.1435	3.487	30.76	0.8776
$+\mathcal{L}_{\text{Reblur, n2}}$	**0.1252**	**2.918**	30.46	0.8717
SRN ($\mathcal{L}_1$ only)	0.1148	3.392	31.89	0.8999
$+\mathcal{L}_{\text{Reblur, n1}}$	0.1071	3.305	32.01	0.9044
$+\mathcal{L}_{\text{Reblur, n2}}$	**0.0947**	**2.875**	31.82	0.9026
DHN ($\mathcal{L}_1$ only)	0.0942	3.288	32.65	0.9152
$+\mathcal{L}_{\text{Reblur, n1}}$	0.0931	3.248	32.57	0.9143
$+\mathcal{L}_{\text{Reblur, n2}}$	**0.0805**	**2.830**	32.44	0.9122

Table 5.3: **Quantitative comparison on REDS [71] dataset by loss function.** The reblurring loss improves LPIPS and NIQE over standard $\mathcal{L}_1$ loss.

the training loss functions. With $\mathcal{L}_{\text{Reblur, n1}}$, LPIPS and NIQE improves to a moderate degree while PSNR and SSIM metrics remain at a similar level. Meanwhile, $\mathcal{L}_{\text{Reblur, n1}}$ more aggressively optimizes the perceptual metrics. The perceptual metric improvements are consistently witnessed with different architectures on both the GOPRO and the REDS dataset.

(a) B (b) $\mathcal{L}_1$ (c) $\mathcal{L}_{\text{VGG}}$ (d) $\mathcal{L}_{\text{Adv}}$ (e) $\mathcal{L}_{\text{Reblur, n1}}$ (f) $\mathcal{L}_{\text{Reblur, n2}}$

Figure 5.5: **Visual comparison of deblurred results by training loss function on GOPRO dataset. Upper:** SRN, **Lower:** U-Net.

5.4.2 Effect of Sharpness Preservation Loss

In training $\mathcal{M}_{\text{R}}$, we used both the blur reconstruction loss $\mathcal{L}_{\text{Blur}}$ and the sharpness preservation loss $\mathcal{L}_{\text{Sharp}}$. The latter term $\mathcal{L}_{\text{Sharp}}$ plays an essential role in concentrating only on the motion-driven blur in the given image and keeping sharp image sharp. Table 5.4 presents the performance gains from using $\mathcal{L}_{\text{Sharp}}$ jointly with $\mathcal{L}_{\text{Blur}}$ in terms of the perceptual quality.

Table 5.4 also justifies the effectiveness of the pseudo-sharp image $\hat{S}$ in sharpness preservation. We found the using with S for $\mathcal{L}_{\text{Sharp}}$ in addition to $\mathcal{L}_{\text{Blur}}$ causes less stable training than using $\hat{S}$. Using the pseudo-sharp image confines the input distribution of $\mathcal{M}_{\text{R}}$ to the output domain of $\mathcal{M}_{\text{D}}$. While the real sharp data S differ from the deblurred image L in terms of realness, the pseudo-sharp image $\hat{S}$ only differs by the sharpness. Thus the reblurring module can focus on the image sharpness without being distracted by other unintended properties. Furthermore, it leads the two loss terms $\mathcal{L}_{\text{Blur}}$ and $\mathcal{L}_{\text{Sharp}}$ to reside under the same objective, amplifying any noticeable blur and keeping sharpness when blur is not found.

Method	LPIPS$_{\downarrow}$	NIQE$_{\downarrow}$	PSNR$^{\uparrow}$	SSIM$^{\uparrow}$
U-Net ($\mathcal{L}_{\text{Blur}}$ only)	0.1301	5.132	29.47	0.8839
$+\mathcal{L}_{\text{Sharp}}$ with S	0.1410	5.307	29.15	0.8694
$+\mathcal{L}_{\text{Sharp}}$ with $\hat{S}$	**0.1238**	**5.124**	29.44	0.8824

Table 5.4: **The effect of the sharpness preservation in training our reblurring module measured on GOPRO [72] dataset.** In (5.2), using the pseudo-sharp image $\hat{S}$ instead of the real one S leads to better deblurring performance. We note that the reblurring module is constructed using 2 ResBlocks.

Method	LPIPS$_{\downarrow}$	NIQE$_{\downarrow}$	PSNR$^{\uparrow}$	SSIM$^{\uparrow}$
SRN ($\mathcal{L}_1$)	0.1246	5.252	30.62	0.9078
$+0.001\mathcal{L}_{\text{Adv}}$	0.1141	4.960	30.53	0.9068
$+0.3\mathcal{L}_{\text{VGG}}$	**0.1037**	4.945	30.60	0.9074
$+\mathcal{L}_{\text{Reblur, n2}}$	**0.1037**	**4.887**	30.57	0.9074

Table 5.5: **Comparison of reblurring loss and other perceptual losses on GOPRO [72] dataset applied to SRN.**

5.4.3 Comparison with Other Perceptual Losses

The reblurringl loss provides a conceptually different learning objectives from the adversarial and the perceptual losses and is designed to focus on the motion blur. Table 5.5 compares the effectiveness of $\mathcal{L}_{\text{Reblur}}$ with adversarial loss $\mathcal{L}_{\text{Adv}}$, and the VGG perceptual loss [42] by applying them to SRN [104] on GOPRO dataset. While our method provides quantitatively better perceptual scores, the different perceptual losses are oriented to varying goals and are not in essentially competing relation. They do not necessarily conflict with each other and can be jointly applied in training to catch the perceptual quality in varying aspects.

5.4.4 Effect of Test-time Adaptation

We conduct test-time adaptation with the proposed self-supervised reblurring loss, $\mathcal{L}^{\text{self}}_{\text{Reblur}}$ to make the deblurred image even sharper. Figure 5.6 shows the test-time-adapted result with SRN. Compared with the baseline trained with L1 loss, our results exhibit improved trade-off relation between PSNR and the perceptual metrics, LPIPS

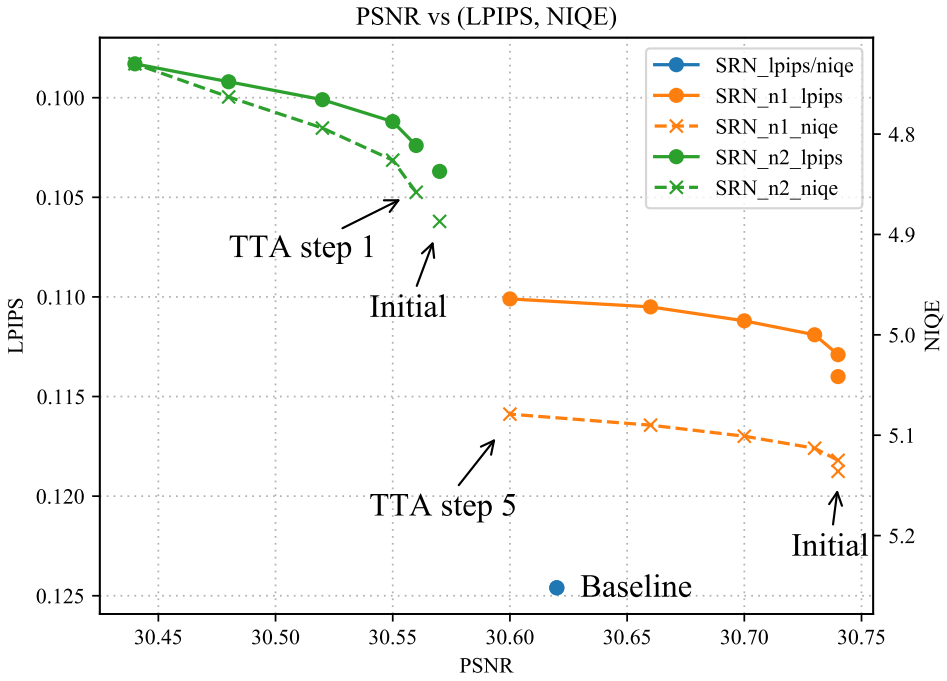


Figure 5.6: **Test-time adaption results using SRN on GOPRO [72] dataset.** The proposed self-supervised objective improves trade-off between the perceptual image quality (LPIPS, NIQE) and PSNR compared with the baseline.

and NIQE. Table 5.6 and 5.7 provide detailed quantitative test-time adaptation results on GOPRO and REDS dataset, respectively with various deblurring module architectures.

5.4.5 Comparison with State-of-The-Art Methods

We have improved the perceptual quality of the deblurred images by training several different model architectures. We compare the perceptual quality with the other state-of-the-art methods in Figure 5.8. Especially, DeblurGAN-v2 was trained with the VGG loss and the adversarial loss. Our results achieve visually sharper texture from the reblurring loss and test-time adaptation.

Method	LPIPS$_{\downarrow}$	NIQE$_{\downarrow}$	PSNR$^{\uparrow}$	SSIM$^{\uparrow}$
U-Net ($\mathcal{L}_1$)	0.1635	5.996	29.66	0.8874
U-Net ($\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n1}}$)	0.1365	5.629	29.58	0.8869
+ TTA step 5	**0.1327**	**5.599**	29.52	0.8878
U-Net ($\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n2}}$)	0.1238	5.124	29.44	0.8824
+ TTA step 5	**0.1187**	**5.000**	29.42	0.8831
SRN ($\mathcal{L}_1$)	0.1246	5.252	30.62	0.9078
SRN ($\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n1}}$)	0.1140	5.136	30.74	0.9104
+ TTA step 1	0.1129	5.125	30.74	0.9107
+ TTA step 2	0.1119	5.113	30.73	0.9108
+ TTA step 3	0.1112	5.101	30.70	0.9108
+ TTA step 4	0.1105	5.090	30.66	0.9105
+ TTA step 5	**0.1101**	**5.079**	30.60	0.9100
SRN ($\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n2}}$)	0.1037	4.887	30.57	0.9074
+ TTA step 5	**0.0983**	**4.730**	30.44	0.9067
DHN ($\mathcal{L}_1$)	0.1179	5.490	31.53	0.9207
DHN ($\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n1}}$)	0.0975	5.472	31.53	0.9217
+ TTA step 5	**0.0940**	**5.343**	31.32	0.9208
DHN ($\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n2}}$)	0.0837	5.076	31.34	0.9177
+ TTA step 5	**0.0805**	**4.948**	31.28	0.9174

Table 5.6: **Test-time adaptation results of various deblurring networks on GOPRO [72] dataset.**

Method	LPIPS$_{\downarrow}$	NIQE$_{\downarrow}$	PSNR$^{\uparrow}$	SSIM$^{\uparrow}$
U-Net ($\mathcal{L}_1$)	0.1486	3.649	30.80	0.8772
U-Net ($\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n2}}$)	0.1252	2.918	30.46	0.8717
+ TTA step 5	**0.1226**	**2.849**	30.25	0.8701
SRN ($\mathcal{L}_1$)	0.1148	3.392	31.89	0.8999
SRN ($\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n2}}$)	0.0947	2.875	31.82	0.9026
+ TTA step 5	**0.0909**	**2.798**	31.50	0.9008
DHN ($\mathcal{L}_1$)	0.0942	3.288	32.65	0.9152
DHN ($\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n2}}$)	0.0805	2.830	32.44	0.9122
+ TTA step 5	**0.0763**	**2.761**	32.17	0.9110

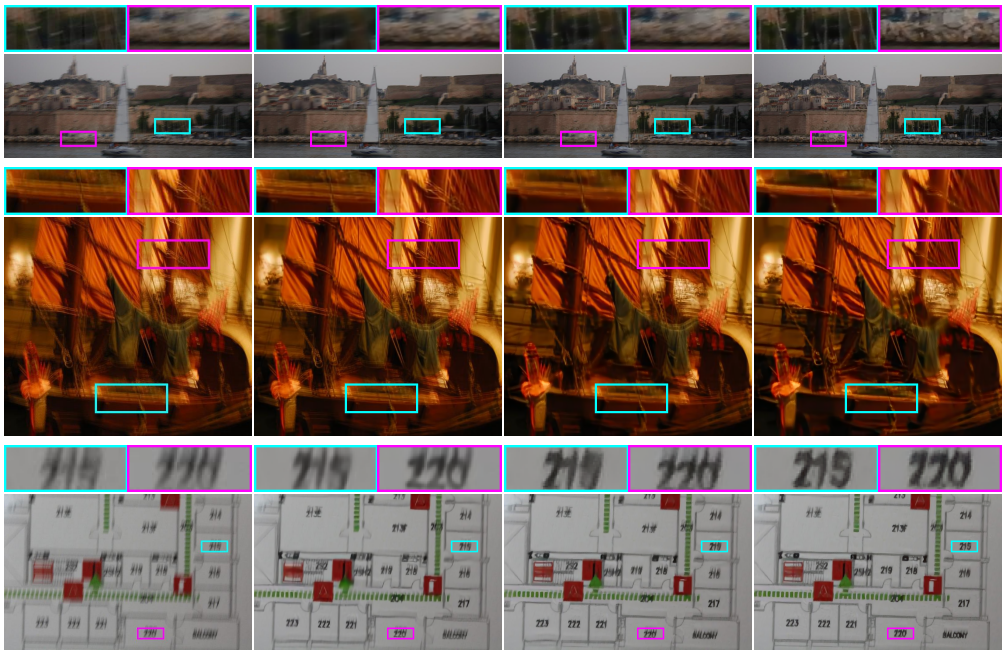
Table 5.7: **Test-time adaptation results of various deblurring methods on REDS [71] dataset.**

(a) B (b) $\mathcal{L}_1$ (c) $\mathcal{L}_{\text{Reblur, n2}}$ (d) + TTA step 5

Figure 5.7: **Qualitative comparison between different training objectives and the test-time adaptation.** Patches are sampled from the REDS [71] dataset validation split.

(a) Blurry input B (b) SE-Sharing [28] (c) DeblurGAN-v2 [53] (d) **Ours (TTA step 5)**

Figure 5.8: **Qualitative comparison between state-of-the-art deblurring methods on the GOPRO [72] dataset.** Our approach uses the SRN [104] model as a baseline architecture.



(a) Blurry input B (b) $\mathcal{L}_1$ (c) $\mathcal{L}_{\textbf{Reblur, n2}}$ **(Ours)** (d) + **TTA step 5 (Ours)**

Figure 5.9: **Qualitative comparison of deblurring results on the real-world images [54] by different loss functions and test-time adaptation.** The proposed test-time adaptation greatly improves visual quality and sharpness of the deblurred images.

5.4.6 Real World Image Deblurring

While our method uses synthetic datasets [72, 71] for training, the trained models generalize to real blurry images. In Figure 5.9, we show deblurred results from Lai *et al.* [54] dataset with DHN model. Compared with the baseline $\mathcal{L}_1$ loss, our reblurring loss $\mathcal{L}_{Reblur,n2}$ provides an improved deblurring quality. As the real test image could deviate from the training data distribution, a single forward inference may not produce optimal results. With the self-supervised test-time adaptation, our deblurred images reveal sharper and detailed textures.

Method	LPIPS$_{\downarrow}$	NIQE$_{\downarrow}$	PSNR$^{\uparrow}$	SSIM$^{\uparrow}$
SRN ($\mathcal{L}_1$ only)	0.1246	5.252	30.62	0.9078
$+\mathcal{L}_{\text{VGG}}$	0.1037	4.945	30.60	0.9074
$+\mathcal{L}_{\text{VGG}} + \mathcal{L}_{\text{Reblur, n2}}$	**0.0928**	**4.671**	30.64	0.9079
$+\mathcal{L}_{\text{Adv}}$	0.1141	4.960	30.53	0.9068
$+\mathcal{L}_{\text{Adv}} + \mathcal{L}_{\text{Reblur, n2}}$	**0.1014**	**4.811**	30.56	0.9075
DHN ($\mathcal{L}_1$ only)	0.1179	5.490	31.53	0.9207
$+\mathcal{L}_{\text{VGG}}$	0.0994	5.022	31.48	0.9195
$+\mathcal{L}_{\text{VGG}} + \mathcal{L}_{\text{Reblur, n2}}$	**0.0773**	**4.897**	31.28	0.9161
$+\mathcal{L}_{\text{Adv}}$	0.0969	5.026	31.46	0.9188
$+\mathcal{L}_{\text{Adv}} + \mathcal{L}_{\text{Reblur, n2}}$	**0.0835**	**4.799**	31.28	0.9162

Table 5.8: **Results on GOPRO [72] dataset by adding reblurring loss to the other preceptual losses.**

Method	LPIPS$_{\downarrow}$	NIQE$_{\downarrow}$	PSNR$^{\uparrow}$	SSIM$^{\uparrow}$
SRN ($\mathcal{L}_1$ only)	0.1148	3.392	31.89	0.8999
$+\mathcal{L}_{\text{VGG}}$	0.1000	3.256	31.86	0.9001
$+\mathcal{L}_{\text{VGG}} + \mathcal{L}_{\text{Reblur, n2}}$	**0.0868**	**2.835**	31.83	0.9015
DHN ($\mathcal{L}_1$ only)	0.0942	3.288	32.65	0.9152
$+\mathcal{L}_{\text{VGG}}$	0.0812	3.171	32.61	0.9146
$+\mathcal{L}_{\text{VGG}} + \mathcal{L}_{\text{Reblur, n2}}$	**0.0723**	**2.821**	32.48	0.9133

Table 5.9: **Results on REDS [71] dataset by adding reblurring loss to the other preceptual losses.**

5.4.7 Combining Reblurring Loss with Other Perceptual Losses

we describe the different characteristics of the proposed reblurring loss and the other perceptual losses. Thus, we can combine our reblurring loss with the other perceptual losses to take advantage in multiple perspectives. Our reblurring loss is a new perceptual loss that is sensitive to blurriness of an image, a type of image structure-level information while other perceptual losses such as VGG loss [42] and adversarial loss [56] are more related to the high-level contexts. As VGG model [95] is trained to recognize image classes, optimizing with VGG loss could make an image better recognizable. In the GAN frameworks [30], it is well known that discriminators can easily tell fake images from real images [109], being robust against JPEG compression and

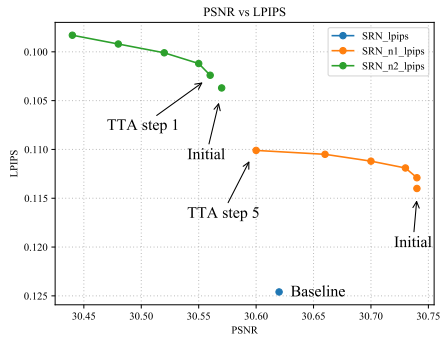
blurring. In the adversarial loss from the discriminator, the realism difference could be more salient than other features such as blurriness.

With the perceptual loss functions designed with different objectives, combining them could bring visual quality improvements in various aspects. Table 5.8 and 5.8 shows the effect of applying our reblurring loss jointly with the other perceptual losses on GOPRO and REDS datasets. We omit the loss coefficients for simplicity. We used weight 0.3 for the VGG loss $\mathcal{L}_{\text{VGG}}$ and 0.001 for the adversarial loss, $\mathcal{L}_{\text{Adv}}$. We witness LPIPS and NIQE further improves when our reblurring loss is combined with $\mathcal{L}_{\text{VGG}}$ or $\mathcal{L}_{\text{Adv}}$.

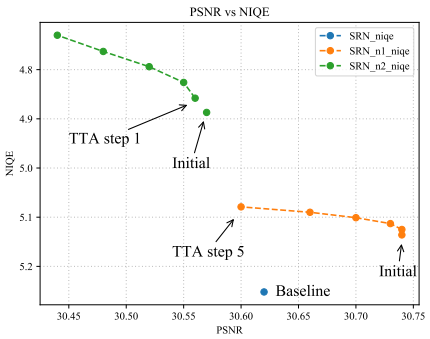
5.4.8 Perception vs. Distortion Trade-Off

We show the effect of test-time adaptation and show the trade-off relation between the conventional distortion quality metric (PSNR, SSIM) and the perceptual metrics (LPIPS, NIQE) compared with the baselines. It is known in image restoration literature that the distortion error and the perceptual quality error are in trade-off relation [7, 6]. The relation is often witnessed by training a single model with different loss functions. In most cases, to obtain a better perceptual quality from a single model architecture, retraining with another loss from scratch is necessary. Our test-time adaptation from self-supervised reblurring loss, in contrast, can provide the steps toward perceptual quality without full retraining.

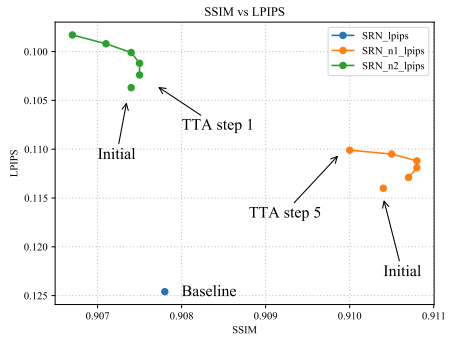
In Figure 5.10 and 5.11, we present the perception-distortion trade-off from our test-time adaption. LPIPS and NIQE scores consistently improve from each adaptation step in both SRN and DHN models. While PSNR is moderately sacrificed from the adaptation, SSIM improves in the early steps as it more reflects the structural information. Our results show improved trade-off between the distortion and perception metrics over the baseline models trained with L1 loss.



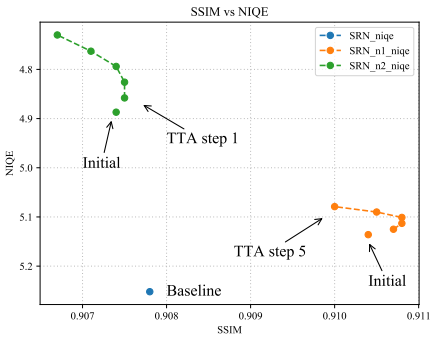
(a) PSNR vs LPIPS



(b) PSNR vs NIQE



(c) SSIM vs LPIPS

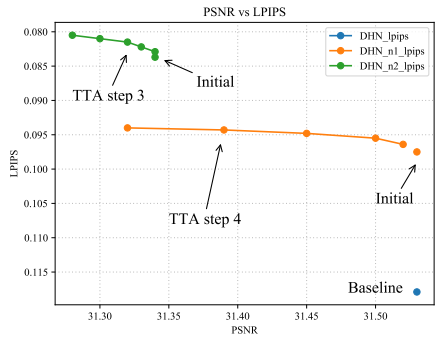


(d) SSIM vs NIQE

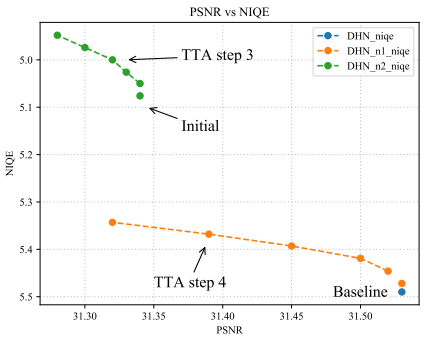
Figure 5.10: **Perception-distortion trade-off from test-time adaptation applied to SRN model on GOPRO [72] dataset.**

5.4.9 Visual Comparison of Loss Function

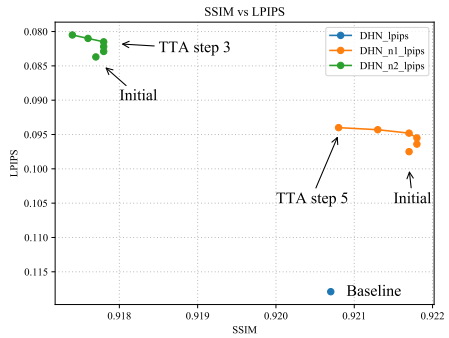
We visually validate the effect of reblurring loss compared with the other perceptual losses. In Figure 5.12 and 5.13, we perform visual ablation by showing the deblurred results from baseline L1 loss, our reblurring loss, and additional test-time adaptation. For $\mathcal{M}_{\mathrm{D}}$, we used DHN. For $\mathcal{M}_{\mathrm{R}}$, 2 ResBlocks are used. Our final result reveals sharper image structure and texture. In Figure 5.14 and 5.15, we compare the effect of 3 different perceptual losses. The results from VGG loss, adversarial loss, and our reblurring loss are shown. Our reblurring loss exhibits clearer edges and the face details than other perceptual losses.



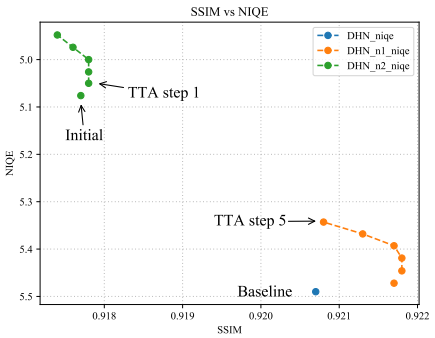
(a) PSNR vs LPIPS



(b) PSNR vs NIQE



(c) SSIM vs LPIPS



(d) SSIM vs NIQE

Figure 5.11: **Perception-distortion trade-off from test-time adaptation applied to DHN model on GOPRO [72] dataset.**

5.4.10 Implementation Details

Model Architecture. In the main manuscript, we performed the experiments with 3 model architectures. We set our baseline model as a light-weight residual U-Net architecture that runs in fast speed. The baseline model is used to design our reblurring loss with pseudo-sharp images through ablation study in Table 5.4.

For reblurring operation, we use a simple residual network $\mathcal{M}_{\text{R}}$ without strides to avoid deconvolution artifacts. The baseline U-Net and the reblurring module architectures are shown in Figure 5.16. The detailed parameters for U-Net and $\mathcal{M}_{\text{R}}$ are each specified in Table 5.10 and 5.12.

(a) Blur (b) Our deblurred image (TTA step 5)

(c) Blur B (d) $\mathcal{L}_1$ (e) $\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n2}}$ (f) Ours (TTA step 5)

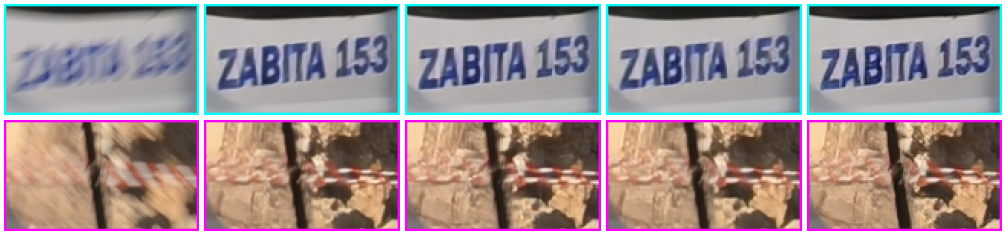
Figure 5.12: **Visual comparison of deblurred results by reblurring loss and test-time adaptation on REDS [71] dataset.**

(a) Blur (b) Our deblurred image (TTA step 5)

(c) Blur B (d) $\mathcal{L}_1$ (e) $\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n2}}$ (f) Ours (TTA step 5)

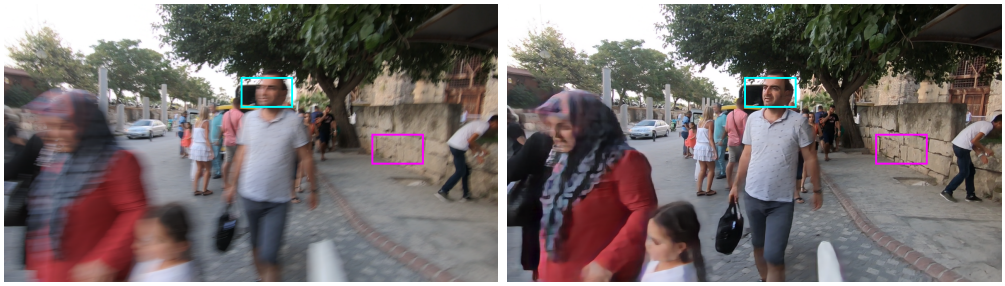
Figure 5.13: **Visual comparison of deblurred results by reblurring loss and test-time adaptation on REDS [71] dataset.**

(a) Blur (b) Our deblurred image ($\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n2}}$)



(c) Blur B (d) $\mathcal{L}_1$ (e) $\mathcal{L}_1 + \mathcal{L}_{\text{VGG}}$ (f) $\mathcal{L}_1 + \mathcal{L}_{\text{Adv}}$ (g) $\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n2}}$

Figure 5.14: **Visual comparison of perceptual losses on REDS [71] dataset.**



(a) Blur (b) Our deblurred image ($\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n2}}$)

(c) Blur B (d) $\mathcal{L}_1$ (e) $\mathcal{L}_1 + \mathcal{L}_{\text{VGG}}$ (f) $\mathcal{L}_1 + \mathcal{L}_{\text{Adv}}$ (g) $\mathcal{L}_1 + \mathcal{L}_{\text{Reblur, n2}}$

Figure 5.15: **Visual comparison of perceptual losses on REDS [71] dataset.**

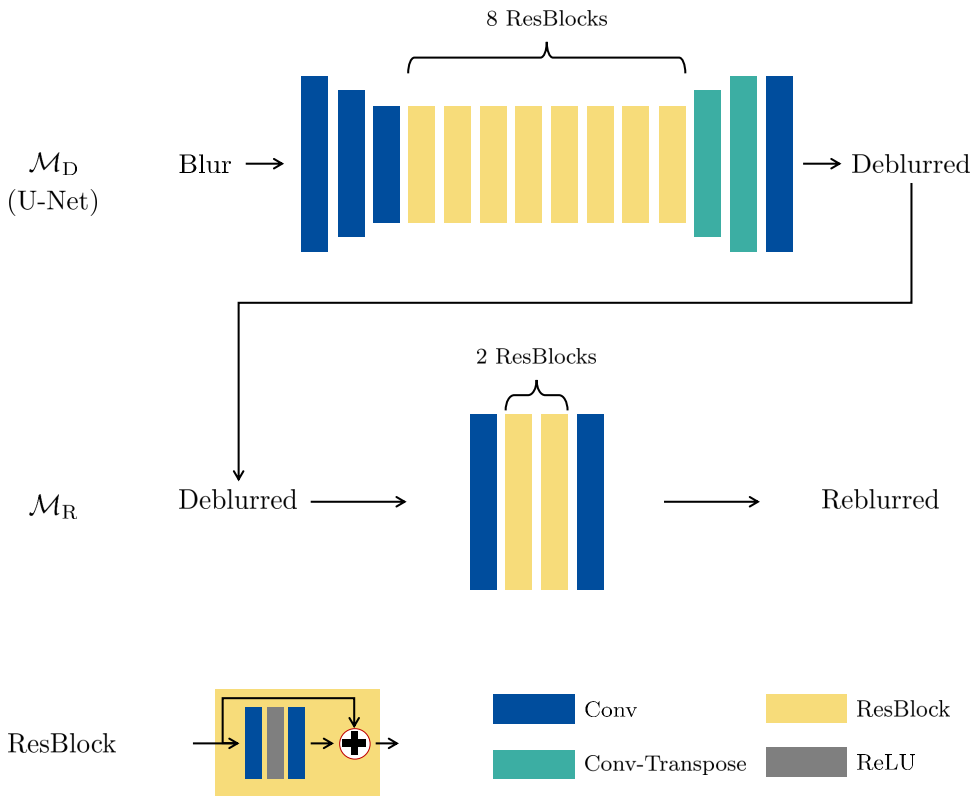


Figure 5.16: **The baseline U-Net architecture and the reblurring module architecture.** We use the same reblurring module for all experiments except for the varying number of ResBlocks.

In addition to the U-Net, experiments were conducted with state-of-the-art deblurring models based on SRN [103] and DMPHN [119]. SRN [104] was originally designed to operate on grayscale images with a LSTM module. Later, the authors released the sRGB version code without LSTM, exhibiting an improved accuracy. We adopted the revised SRN structure in our experiments.

The other model we chose is based on DMPHN (1-2-4-8) [119]. DMPHN performs hierarchical residual refinement to produce the final output. The model consists of convolutional layers with ReLU activations that are spatially shift-equivariant. In [119], each level splits the image and performs convolutional operation on the divided

#	Layer description	Output shape
	Input	$3 \times H \times W$
1	5×5 conv	$64 \times H \times W$
2	3×3 conv	$128 \times H/2 \times W/2$
3	3×3 conv	$192 \times H/4 \times W/4$
4-19	8 ResBlocks (3×3)	$192 \times H/4 \times W/4$
20	3×3 conv	$128 \times H/2 \times W/2$
21	3×3 conv	$64 \times H \times W$
22	5×5 conv	$3 \times H \times W$

Table 5.10: **U-Net module specifics**

Method	LPIPS$_{\downarrow}$	NIQE$_{\downarrow}$	PSNR$^{\uparrow}$	SSIM$^{\uparrow}$
DMPHN ($\mathcal{L}_1$ only)	0.1184	5.542	31.42	0.9191
DHN ($\mathcal{L}_1$ only)	**0.1179**	**5.490**	**31.53**	**0.9207**

Table 5.11: **DMPHN modification results on GOPRO [72] dataset.** DHN without patch-wise convolution brings improved accuracy.

patches. As the convolutional weights do not differ by the patches, the operations do not necessarily have to be done patch-wise. Thus, we remove the multi-patch strategy and perform the convolution on the whole given input without dividing the image into patches. We refer to the modified model as DHN. As shown in Table 5.11, convolution on the whole image instead of patch-wise convolution brings higher accuracy.

Metrics To quantitatively compare the deblurred images in the following sections, we use PSNR, SSIM [110], LPIPS [126], and NIQE [70]. In the image deblurring literature, SSIM has been measured by MATLAB `ssim` function on sRGB images with $H \times W \times C$. SSIM was originally developed for grayscale images and MATLAB `ssim` function for a 3-dimensional tensor considers an image to be a 3D grayscale volume image. Thus, most of the previous SSIM measures were not accurate, leading to higher values. Instead, we measured all the SSIM for each channel separately and averaged them. We used `skimage.metrics.structural_similarity` function in the scikit-image package for python to measure SSIM for multi-channel images.

Training For all the experiments, we performed the same training process for a fair

#	Layer description	Output shape
	Input	$3 \times H \times W$
1	5×5 conv	$64 \times H \times W$
2-5	2 ResBlocks (5×5)	$64 \times H/4 \times W/4$
6	5×5 conv	$3 \times H \times W$

Table 5.12: **Reblurring module specifics**

comparison. On the GOPRO dataset [72], we trained each model for 4000 epochs. On the REDS dataset [71], the models are trained for 200 epochs. Adam [49] optimizer is used in all cases. When calculating distance between images with Lp norm, we always set $p = 1$, using L1 distance. Starting from the initial learning rate 1×10^{-4}, the learning rate halves when training reaches 50%, 75%, and 90% of the total epochs. We used PyTorch 1.7.1 with CUDA 11.0 to implement the deblurring methods. Mixed-precision training [69] is employed to accelerate operations on RTX 2080 Ti GPUs.

5.4.11 Determining Reblurring Module Size

We describe how the reblurring module design and the size are determined. As our reblurring loss $\mathcal{L}_{\mathrm{R}}$ is realized by $\mathcal{M}_{\mathrm{R}}$, the reblurring module design plays an essential role. As shown in Figure 5.16, the $\mathcal{M}_{\mathrm{R}}$ architecture is a simple ResNet. Table 5.13 shows the relation between the deblurring performance and $\mathcal{M}_{\mathrm{R}}$ size by changing the number of ResBlocks.

For all deblurring module $\mathcal{M}_{\mathrm{D}}$ architectures, LPIPS was the best when the number of ResBlocks, $n = 2$. NIQE showed good performance when $2 \leq n \leq 3$. PSNR and SSIM had tendency to decrease when $n \geq 1$. For larger number of ResBlocks, we witnessed sharper edges could be obtained but sometimes, cartoon artifacts with over-strong edges were witnessed.

Considering the trade-off between the PSNR and the perceptual metrics, we chose $n \in \{1, 2\}$ in the following experiments. $n = 1$ finds balance between the PSNR and LPIPS and $n = 2$ puts more weight on the perceptual quality.

Method	LPIPS$_{\downarrow}$	NIQE$_{\downarrow}$	PSNR$^{\uparrow}$	SSIM$^{\uparrow}$
U-Net ($\mathcal{L}_1$ only)	0.1635	5.996	**29.66**	**0.8874**
$+\mathcal{L}_{\text{Reblur, n1}}$	0.1365	5.629	29.58	0.8869
$+\mathcal{L}_{\text{Reblur, n2}}$	**0.1238**	**5.124**	29.44	0.8824
$+\mathcal{L}_{\text{Reblur, n3}}$	0.1386	5.448	29.38	0.8819
$+\mathcal{L}_{\text{Reblur, n4}}$	0.1415	5.513	29.25	0.8789
SRN ($\mathcal{L}_1$ only)	0.1246	5.252	30.62	0.9078
$+\mathcal{L}_{\text{Reblur, n1}}$	0.1140	5.136	**30.74**	**0.9104**
$+\mathcal{L}_{\text{Reblur, n2}}$	**0.1037**	4.887	30.57	0.9074
$+\mathcal{L}_{\text{Reblur, n3}}$	0.1091	**4.875**	30.50	0.9060
$+\mathcal{L}_{\text{Reblur, n4}}$	0.1155	5.041	30.53	0.9056
DHN ($\mathcal{L}_1$ only)	0.1179	5.490	**31.53**	0.9207
$+\mathcal{L}_{\text{Reblur, n1}}$	0.0975	5.472	**31.53**	**0.9217**
$+\mathcal{L}_{\text{Reblur, n2}}$	**0.0837**	5.076	31.34	0.9177
$+\mathcal{L}_{\text{Reblur, n3}}$	0.0845	**4.963**	31.26	0.9159
$+\mathcal{L}_{\text{Reblur, n4}}$	0.0861	5.041	31.19	0.9149

Table 5.13: **The effect of reblurring loss on GOPRO [72] dataset by the reblurrimg module size.** Reblurring module size varies by the number of ResBlocks.

5.5 Conclusion

In this paper, we validate a new observation that clean sharp images are hard to reblur and develop new low-level perceptual loss. We construct reblurring loss that cares for the image blurriness by jointly training a pair of deblurring and reblurring modules. The supervised reblurring loss provides an amplified view on motion blur while the self-supervised loss inspects the blurriness from the learned reblurring module weights. The self-supervision lets the deblurring module adapt to the new image at test time without ground truth. By applying the loss terms to the state-of-the-art deblurring architectures, we demonstrated our method consistently improves the the perceptual sharpness of the deblurred images both quantitatively and visually.

Chapter 6

Conclusion

In this dissertation, deep-learning based dynamnic scene deblurring methods are proposed by addressing the problems of components in the learning process. The large-scale datasets for training and evaluation are constructed, suitable neural network architectures were designed for image and video deblurring, and the loss funciton is jointly learned in the training procedure to facilitate deblurring in a higher-quality.

In Chapter 2, the first dynamic scene deblurring dataset, GOPRO is constructed from high-speed videos. GOPRO provides the realistic blurry images as well as the paired sharp images. By using the temporally center-aligned image pairs, neural networks models can be trained with supervision. Furthermore, an improved REDS dataset is constructed by elevating the quality of the ground-truth sharp images as well as the realism in motion blur. Other practically following artifacts such as compression and low-resolution are jointly considered in the variations. Both GOPRO and REDS serve as standard benchmarks in image and video deblurring field of research. They are widely used for research and public competition series have been organized.

In Chapter 3, the first end-to-end neural network architecture for dynamic scene deblurring is proposed. Adopting deep learning avoids the complex blur trajectory formulation and let the neural network automatically find the deblurred image directly. The model is trained with the GOPRO and the REDS datasets in end-to-end fash-

ion without having to estimate the blur kernel explicitly. Henceforth, the deblurred images do not suffer from the kernel-related errors such as ringing artifacts without using handcrafted prior knowledge. Furthermore, the multi-scale architecture inspired by the coarse-to-fine optimization effectively enlarges the receptive field, improving both the inference time and the deblurring accuracy. The learned deblurring operation generalizes to real blurry images.

In Chapter 4, recurrent neural networks with intra-frame iterations are proposed for video deblurring. While the basic RNNs lets the hidden state propagate the information in the past frames to the future frames, intra-frame iterations tries to better exploit the received knowledge. Intra-frame iteration reuses the recurrence relation on the given target frame to update the initial hidden state. As the initial hidden state only contains information from the past frames, recomputing the hidden state with the current frame can better capture the information from both the past and current frames. Henceforth, the proposed intra-frame iterations improve the deblurring performance of recurrent networks without any additional parameters as the existing recurrence relation is reused. In addition to the basic single-cell recurrent architecture, dual-cell architecture is proposed with the second RNN cell specialized for updating the hidden states. To further improve the training of intra-frame iterative architecture, stochastic training is employed with regularization loss. The loss favors additional computations only when significant content accuracy improvements are witnessed. Beginning from a light-weight baseline, the proposed IFI-RNNs achieve state-of-the-art deblurring performance as well as fast inference speed.

In Chapter 5, a novel loss function for deblurring is designed from the joint learning. A new observation is obtained that an imperfectly deblurred image still contains partial blur information such that original blur can be reconstructed while a ground-truth clean images don't. Motivated from the new clue, a reblurring module is jointly trained along with the deblurring module to make the deblurred images hard to reblur. From the learned reblurring module, reblurring loss is designed in both supervised and

self-supervised forms. The supervised reblurring loss compares the amplified blurs of the deblurred and the sharp image. The self-supervised reblurring loss penalizes the remaining blur trace by inspecting if the image is changed from reblurring. As the self-supervised loss does not require ground-truth, it is used at test-time to let the deblurring module adapt to the given image. Experimental results demonstrate that the models trained with reblurring losses exhibit sharper images with improved perceptual quality.

While the proposed approaches investigates the training of deep neural networks for deblurring, there still remains research issues.

- Photographies and videos are typically captured on hand-held devices in real time where motion blur are prone to occur. Compared with the capturing speed, state-of-the-art deblurring methods at current state requires execution at a workstation with high-performance GPUs for a few seconds. In order for the deblurring methods to be applicable in practice, the inference stage should be accelerated. Often, neural networks are quantized and pruned to relieve the computation burden but such techniques have not been much studied in image restoration fields due to the sensitivity in the restoration accuracy. To address this practical issue, network acceleration techniques will be studied while preserving the state-of-the-art deblurring accuracy as much as possible.

- Deep learning has shown impressive results in computer vision but often generalization to unseen data is an issue. Collecting massive amount of data to better train models could be a solution but this is not always possible. Specifically, deblurring may require manual capturing of diverse scenes with multiple cameras which is labor intensive. On the other hand, unsupervised learning could relieve the data collection burden by using unpaired set of the blurry and the sharp images. While the supervised and self-supervised learning has been investigated, unsupervised learning in a large scale can be considered a way for better generalization to unseen types of views and blurs.

Bibliography

[1] Byeongjoo Ahn, Tae Hyun Kim, Wonsik Kim, and Kyoung Mu Lee. Occlusion-aware video deblurring with a new layered blur model. *arXiv preprint arXiv:1611.09572*, 2016.

[2] Miika Aittala and Fredo Durand. Burst image deblurring using permutation invariant convolutional neural networks. In *ECCV*, 2018.

[3] Soonmin Bae and Frédo Durand. Defocus magnification. *Computer Graphics Forum*, 26(3):571–579, 2007.

[4] Yuval Bahat, Netalee Efrat, and Michal Irani. Non-uniform blind deblurring by reblurring. In *ICCV*, 2017.

[5] Leah Bar, Benjamin Berkels, Martin Rumpf, and Guillermo Sapiro. A variational framework for simultaneous motion estimation and restoration of motion-blurred video. In *ICCV*, pages 1–8. IEEE, 2007.

[6] Yochai Blau, Roey Mechrez, Radu Timofte, Tomer Michaeli, and Lihi Zelnik-Manor. The 2018 pirm challenge on perceptual image super-resolution. In *Proceedings of the ECCV Workshops*, September 2018.

[7] Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *CVPR*, 2018.

[8] Tim Brooks and Jonathan T. Barron. Learning to synthesize motion blur. In *CVPR*, 2019.

[9] Jian-Feng Cai, Hui Ji, Chaoqiang Liu, and Zuowei Shen. Blind motion deblurring using multiple images. *Journal of Computational Physics*, 228(14):5057–5071, 2009.

[10] Ayan Chakrabarti. A neural approach to blind motion deblurring. In *ECCV*, pages 221–235. Springer, 2016.

[11] Huaijin Chen, Jinwei Gu, Orazio Gallo, Ming-Yu Liu, Ashok Veeraraghavan, and Jan Kautz. Reblur2deblur: Deblurring videos via self-supervised learning. In *ICCP*, 2018.

[12] Sunghyun Cho, Hojin Cho, Yu-Wing Tai, and Seungyong Lee. Registration based non-uniform motion deblurring. In *Computer Graphics Forum*, volume 31, pages 2183–2192. Wiley Online Library, 2012.

[13] Sunghyun Cho and Seungyong Lee. Fast motion deblurring. In *SIGGRAPH Asia*, 2009.

[14] Sunghyun Cho, Yasuyuki Matsushita, and Seungyong Lee. Removing non-uniform motion blur from images. In *ICCV*, pages 1–8. IEEE, 2007.

[15] Sunghyun Cho, Jue Wang, and Seungyong Lee. Video deblurring for hand-held cameras using patch-based synthesis. *ACM TOG*, 31(4):64, 2012.

[16] Florent Couzinie-Devy, Jian Sun, Karteek Alahari, and Jean Ponce. Learning to estimate and remove non-uniform image blur. In *CVPR*, June 2013.

[17] Mauricio Delbracio and Guillermo Sapiro. Burst deblurring: Removing camera shake through fourier burst accumulation. In *CVPR*, 2015.

[18] Mauricio Delbracio and Guillermo Sapiro. Removing camera shake via weighted fourier burst accumulation. *IEEE TIP*, 24(11):3293–3307, 2015.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE, 2009.

[20] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, pages 1486–1494, 2015.

[21] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE TPAMI*, 38(2):295–307, 2015.

[22] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, December 2015.

[23] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, pages 2650–2658, 2015.

[24] David Eigen, Dilip Krishnan, and Rob Fergus. Restoring an image taken through a window covered with dirt or rain. In *ICCV*, December 2013.

[25] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, pages 2366–2374, 2014.

[26] Rob Fergus, Barun Singh, Aaron Hertzmann, Sam T Roweis, and William T Freeman. Removing camera shake from a single photograph. In *SIGGRAPH*, pages 787–794, 2006.

[27] Michael Figurnov, Maxwell D. Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *CVPR*, July 2017.

[28] Hongyun Gao, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Dynamic scene deblurring with parameter selective sharing and nested skip connections. In *CVPR*, 2019.

[29] Dong Gong, Jie Yang, Lingqiao Liu, Yanning Zhang, Ian Reid, Chunhua Shen, Anton van den Hengel, and Qinfeng Shi. From motion blur to motion flow: A deep learning solution for removing heterogeneous motion blur. In *CVPR*, 2017.

[30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[31] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

[32] Ankit Gupta, Neel Joshi, C Lawrence Zitnick, Michael Cohen, and Brian Curless. Single image deblurring using motion density functions. In *ECCV*, 2010.

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

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, June 2016.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[36] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

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

[38] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, pages 646–661. Springer, 2016.

[39] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[40] Hui Ji and Kang Wang. A two-stage approach to blind spatially-varying motion deblurring. In *CVPR*. IEEE, 2012.

[41] Meiguang Jin, Zhe Hu, and Paolo Favaro. Learning to extract flawless slow motion from blurry videos. In *CVPR*, 2019.

[42] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[43] Tae Hyun Kim, Byeongjoo Ahn, and Kyoung Mu Lee. Dynamic scene deblurring. In *ICCV*, 2013.

[44] Tae Hyun Kim and Kyoung Mu Lee. Segmentation-free dynamic scene deblurring. In *CVPR*, 2014.

[45] Tae Hyun Kim and Kyoung Mu Lee. Generalized video deblurring for dynamic scenes. In *CVPR*, 2015.

[46] Tae Hyun Kim, Kyoung Mu Lee, Bernhard Schölkopf, and Michael Hirsch. Online video deblurring via dynamic temporal blending network. In *ICCV*, 2017.

[47] Tae Hyun Kim, Seungjun Nah, and Kyoung Mu Lee. Dynamic video deblurring using a locally adaptive blur model. *IEEE TPAMI*, 40(10):2374–2387, 2017.

[48] Tae Hyun Kim, Mehdi S. M. Sajjadi, Michael Hirsch, and Bernhard Schölkopf. Spatio-temporal transformer network for video restoration. In *ECCV*, 2018.

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

[50] Rolf Köhler, Michael Hirsch, Betty Mohler, Bernhard Schölkopf, and Stefan Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In *ECCV*, pages 27–40. Springer, 2012.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 25:1097–1105, 2012.

[52] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. DeblurGAN: Blind motion deblurring using conditional adversarial networks. In *CVPR*, 2018.

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

[54] Wei-Sheng Lai, Jia-Bin Huang, Zhe Hu, Narendra Ahuja, and Ming-Hsuan Yang. A comparative study for single image blind deblurring. In *CVPR*, 2016.

[55] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *ICLR*, 2017.

[56] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[57] Hee Seok Lee and Kyoung Mu Lee. Dense 3d reconstruction from severely blurred images using a single moving camera. In *CVPR*, pages 273–280, 2013.

[58] Anat Levin. Blind motion deblurring using image statistics. *NIPS*, 2006.

[59] Lerenhan Li, Jinshan Pan, Wei-Sheng Lai, Changxin Gao, Nong Sang, and Ming-Hsuan Yang. Learning a discriminative prior for blind image deblurring. In *CVPR*, 2018.

[60] Yunpeng Li, Sing Bing Kang, Neel Joshi, Steve M Seitz, and Daniel P Huttenlocher. Generating sharp panoramas from motion-blurred videos. In *CVPR*, pages 2424–2431. IEEE, 2010.

[61] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, 2017.

[62] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *ICLR*, 2014.

[63] Ce Liu and Deqing Sun. A bayesian approach to adaptive video super resolution. In *CVPR 2011*, pages 209–216. IEEE, 2011.

[64] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, June 2015.

[65] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[66] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[67] Yasuyuki Matsushita, Eyal Ofek, Weina Ge, Xiaoou Tang, and Heung-Yeung Shum. Full-frame video stabilization with motion inpainting. *IEEE TPAMI*, 28(7):1150–1163, 2006.

[68] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. PULSE: Self-supervised photo upsampling via latent space exploration of generative models. In *CVPR*, 2020.

[69] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[70] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE SPL*, 20(3):209–212, 2012.

[71] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. NTIRE 2019 challenges on video deblurring and super-resolution: Dataset and study. In *CVPR Workshops*, 2019.

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

[73] Seungjun Nah, Sanghyun Son, Jaerin Lee, and Kyoung Mu Lee. Clean images are hard to reblur: A new clue for deblurring. *arXiv preprint arXiv:2104.12665*, 2021.

[74] Seungjun Nah, Sanghyun Son, and Kyoung Mu Lee. Recurrent neural networks with intra-frame iterations for video deblurring. In *CVPR*, 2019.

[75] Seungjun Nah, Sanghyun Son, Suyoung Lee, Radu Timofte, and Kyoung Mu Lee. NTIRE 2021 challenge on image deblurring. In *CVPR Workshops*, June 2021.

[76] Seungjun Nah, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. NTIRE 2020 challenge on image and video deblurring. In *CVPR Workshops*, June 2020.

[77] Seungjun Nah, Radu Timofte, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, and Kyoung Mu Lee. NTIRE 2019 challenge on video deblurring: Methods and results. In *CVPR Workshops*, 2019.

[78] Seungjun Nah, Radu Timofte, Shuhang Gu, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, and Kyoung Mu Lee. NTIRE 2019 challenge on video super-resolution: Methods and results. In *CVPR Workshops*, 2019.

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

[80] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *ICCV*, Oct 2017.

[81] T. M. Nimisha, Akash Kumar Singh, and A. N. Rajagopalan. Blur-invariant deep learning for blind-deblurring. In *ICCV*, 2017.

[82] Mehdi Noroozi, Paramanand Chandramouli, and Paolo Favaro. Motion deblurring in the wild. In *GCPR*, 2017.

[83] Jinshan Pan, Deqing Sun, Hanspeter Pfister, and Ming-Hsuan Yang. Blind image deblurring using dark channel prior. In *CVPR*, 2016.

[84] Liyuan Pan, Yuchao Dai, Miaomiao Liu, and Fatih Porikli. Simultaneous stereo video deblurring and scene flow estimation. In *CVPR*, 2017.

[85] Chandramouli Paramanand and A. N. Rajagopalan. Non-uniform motion deblurring for bilayer scenes. In *CVPR*, pages 1115–1122, 2013.

[86] Dongwon Park, Dong Un Kang, Jisoo Kim, and Se Young Chun. Multi-temporal recurrent neural networks for progressive non-uniform single image deblurring with incremental temporal training. In *ECCV*, 2020.

[87] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[88] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[89] Dongwei Ren, Kai Zhang, Qilong Wang, Qinghua Hu, and Wangmeng Zuo. Neural blind deconvolution using deep priors. In *CVPR*, 2020.

[90] Mark A Robertson, Sean Borman, and Robert L Stevenson. Dynamic range improvement through multiple exposures. In *ICIP*, volume 3, pages 159–163. IEEE, 1999.

[91] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[92] Christian J Schuler, Michael Hirsch, Stefan Harmeling, and Bernhard Schölkopf. Learning to deblur. *IEEE TPAMI*, 38(7):1439–1451, 2015.

[93] Qi Shan, Jiaya Jia, and Aseem Agarwala. High-quality motion deblurring from a single image. *ACM TOG*, 27(3):1–10, 2008.

[94] Ziyi Shen, Wenguan Wang, Xiankai Lu, Jianbing Shen, Haibin Ling, Tingfa Xu, and Ling Shao. Human-aware motion deblurring. In *ICCV*, 2019.

[95] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[96] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[97] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *CVPR*, 2017.

[98] Maitreya Suin, Kuldeep Purohit, and A. N. Rajagopalan. Spatially-attentive patch-hierarchical network for adaptive motion deblurring. In *CVPR*, 2020.

[99] Jian Sun, Wenfei Cao, Zongben Xu, and Jean Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *CVPR*, 2015.

[100] Libin Sun, Sunghyun Cho, Jue Wang, and James Hays. Edge-based blur kernel estimation using patch priors. In *ICCP*, 2013.

[101] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, June 2015.

[102] Yu-Wing Tai, Xiaogang Chen, Sunyeong Kim, Seon Joo Kim, Feng Li, Jie Yang, Jingyi Yu, Yasuyuki Matsushita, and Michael S Brown. Nonlinear camera response functions and image deblurring: Theoretical analysis and practice. *IEEE TPAMI*, 35(10):2498–2512, 2013.

[103] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. Detail-revealing deep video super-resolution. In *ICCV*, 2017.

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

[105] Jacob Telleen, Anne Sullivan, Jerry Yee, Oliver Wang, Prabath Gunawardane, Ian Collins, and James Davis. Synthetic shutter speed imaging. In *Computer Graphics Forum*, volume 26, pages 591–598. Wiley Online Library, 2007.

[106] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, September 2018.

[107] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *NIPS*, pages 550–558, 2016.

[108] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, pages 1058–1066. PMLR, 2013.

[109] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A. Efros. CNN-Generated images are surprisingly easy to spot... for now. In *CVPR*, 2020.

[110] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004.

[111] Oliver Whyte, Josef Sivic, Andrew Zisserman, and Jean Ponce. Non-uniform deblurring for shaken images. *IJCV*, 98(2):168–186, 2012.

[112] Patrick Wieschollek, Michael Hirsch, Bernhard Schölkopf, and Hendrik P. A. Lensch. Learning blind motion deblurring. In *ICCV*, 2017.

[113] Patrick Wieschollek, Bernhard Schölkopf, Hendrik PA Lensch, and Michael Hirsch. End-to-end learning for image burst deblurring. In *ACCV*, 2016.

[114] Jonas Wulff and Michael Julian Black. Modeling blurred video with layers. In *ECCV*, 2014.

[115] Li Xu, Jimmy S Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. *NIPS*, 27:1790–1798, 2014.

[116] Li Xu, Shicheng Zheng, and Jiaya Jia. Unnatural L0 sparse representation for natural image deblurring. In *CVPR*, 2013.

[117] Yuan Yuan, Wei Su, and Dandan Ma. Efficient dynamic scene deblurring using spatially variant deconvolution network with optical flow guided training. In *CVPR*, 2020.

[118] Haichao Zhang and Lawrence Carin. Multi-shot imaging: Joint alignment, deblurring and resolution-enhancement. In *CVPR*, 2014.

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

[120] Haichao Zhang, David Wipf, and Yanning Zhang. Multi-image blind deblurring using a coupled adaptive sparse prior. In *CVPR*, 2013.

[121] Haichao Zhang, David Wipf, and Yanning Zhang. Multi-observation blind deconvolution with an adaptive sparse prior. *IEEE TPAMI*, 36(8):1628–1643, 2014.

[122] Haichao Zhang and Jianchao Yang. Intra-frame deblurring by leveraging inter-frame camera motion. In *CVPR*, June 2015.

[123] Jiawei Zhang, Jinshan Pan, Jimmy Ren, Yibing Song, Linchao Bao, Rynson W.H. Lau, and Ming-Hsuan Yang. Dynamic scene deblurring using spatially variant recurrent neural networks. In *CVPR*, 2018.

[124] Kaihao Zhang, Wenhan Luo, Yiran Zhong, Lin Ma, Bjorn Stenger, Wei Liu, and Hongdong Li. Deblurring by realistic blurring. In *CVPR*, 2020.

[125] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE TIP*, 26(7):3142–3155, 2017.

[126] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[127] Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, 2011.

국문 초록

사진 촬영의 궁극적인 목표는 고품질의 깨끗한 영상을 얻는 것이다. 현실적으로, 일상의 사진은 자주 흔들린 카메라와 움직이는 물체가 있는 동적 환경에서 찍는다. 노출시간 중의 카메라와 피사체간의 상대적인 움직임은 사진과 동영상에서 모션 블러를 일으키며 시각적인 화질을 저하시킨다. 동적 환경에서 블러의 세기와 움직임의 모양은 매 이미지마다, 그리고 매 픽셀마다 다르다. 국지적으로 변화하는 블러의 성질은 사진과 동영상에서의 모션 블러 제거를 심각하게 풀기 어려우며 해답이 하나로 정해지지 않은, 잘 정의되지 않은 문제로 만든다.

물리적인 움직임 모델링을 통해 해석적인 접근법을 설계하기보다는 머신러닝 기반의 접근법은 이러한 잘 정의되지 않은 문제를 푸는 보다 현실적인 답이 될 수 있다. 특히 딥 러닝은 최근 컴퓨터 비전 학계에서 표준적인 기법이 되어 가고 있다. 본 학위논문은 사진 및 비디오 디블러링 문제에 대해 딥 러닝 기반 솔루션을 도입하며 여러 현실적인 문제를 다각적으로 다룬다.

첫 번째로, 디블러링 문제를 다루기 위한 데이터셋을 취득하는 새로운 방법을 제안한다. 모션 블러가 있는 이미지와 깨끗한 이미지를 시간적으로 정렬된 상태로 동시에 취득하는 것은 쉬운 일이 아니다. 데이터가 부족한 경우 디블러링 알고리즘들을 평가하는 것 뿐만 아니라 지도학습 기법을 개발하는 것도 불가능해진다. 그러나 고속 비디오를 사용하여 카메라 영상 취득 파이프라인을 모방하면 실제적인 모션 블러 이미지를 합성하는 것이 가능하다. 기존의 블러 합성 기법들과 달리 제안하는 방법은 여러 움직이는 피사체들과 다양한 영상 깊이, 움직임 경계에서의 가리워짐 등으로 인한 자연스러운 국소적 블러의 복잡도를 반영할 수 있다.

두 번째로, 제안된 데이터셋에 기반하여 새로운 단일영상 디블러링을 위한 뉴럴 네트워크 구조를 제안한다. 최적화기법 기반 이미지 디블러링 방식에서 널리 쓰이고 있는 점차적 미세화 접근법을 반영하여 다중규모 뉴럴 네트워크를 설계한다. 제안된 다중규모 모델은 비슷한 복잡도를 가진 단일규모 모델들보다 높은 복원 정확도를 보인다.

세 번째로, 비디오 디블러링을 위한 순환 뉴럴 네트워크 모델 구조를 제안한다. 디블러링을 통해 고품질의 비디오를 얻기 위해서는 각 프레임간의 시간적인 정보와 프레임 내부적인 정보를 모두 사용해야 한다. 제안하는 내부프레임 반복적 연산구조는 두 정보를 효과적으로 함께 사용함으로써 모델 파라미터 수를 증가시키지 않고도 디블러 정확도를 향상시킨다.

마지막으로, 새로운 디블러링 모델들을 보다 잘 최적화하기 위해 로스 함수를 제안한다. 깨끗하고 또렷한 사진 한 장으로부터 자연스러운 모션 블러를 만들어내는 것은 블러를 제거하는 것과 마찬가지로 어려운 문제이다. 그러나 통상 사용하는 로스 함수로 얻은 디블러링 방법들은 블러를 완전히 제거하지 못하며 디블러된 이미지의 남아있는 블러로부터 원래의 블러를 재건할 수 있다. 제안하는 리블러링 로스 함수는 디블러링 수행시 모션 블러를 보다 잘 제거하도록 설계되었다. 이에 나아가 제안한 자기지도학습 과정으로부터 테스트시 모델이 새로운 데이터에 적응하도록 할 수 있다.

이렇게 제안된 데이터셋, 모델 구조, 그리고 로스 함수를 통해 딥 러닝에 기반하여 단일 영상 및 비디오 디블러링 기법들을 제안한다. 광범위한 실험 결과로부터 정량적 및 정성적으로 최첨단 디블러링 성과를 증명한다.

주요어: 디블러링, 동적 영상, 데이터셋, 이미지, 비디오, 로스, 딥 러닝
학번: 2014-21661

감사의 글

지난 학위기간 동안 저를 지도해 주신 이경무 교수님께 깊은 감사의 말씀을 드립니다. 이경무 교수님은 연구하는 방법을 지도해 주셨을 뿐만 아니라 연구자로서 어떻게 생각하고 행동하는지, 새로운 변화를 어떻게 받아들여야 하는지, 그리고 연구자 커뮤니티의 일원으로 살아가는 모습도 함께 보여주셨습니다. 대학원 연구실이라는 한 공간에서 이렇게 다방면으로 지도를 받을 수 있었던 것은 제게 큰 행운이라고 생각합니다. 언제나 연구에 대한 열정을 갖고 계신 이경무 교수님의 모습이 특히 기억에 남고 앞으로도 연구활동을 계속하면서 가르침을 되새기도록 하겠습니다.

제가 신입생일 때 박사과정을 먼저 거쳐간 선배로서, 국제 무대에서 연구자로서 적극적으로 활동하는 모범을 보여주신 조민수, 권준석 교수님께 감사합니다. 처음으로 참여했던 국제 학회에서 만났을 때 해주신 조언들이 박사과정을 시작하는 기반이 되었습니다. 또 매주 진행되는 비전 세미나 때 날카로운 질문들로 통찰력과 함께 어떻게 비판적인 시각을 연마해야 하는지 보여주신 이수찬 교수님께 감사합니다. 그리고 제가 신입생으로 들어오던 해에 박사학위를 마치고 졸업하시던 김원식, 신영민 선배들께 감사합니다. 함께 있었던 기간은 짧지만 친절히 저를 일원으로 맞아주셨고, 특히 김원식 선배는 식사시간까지 아껴 가며 제 질문에 답해주셨던 모습이 기억에 남습니다. 특히 조민수, 이수찬, 김원식 선배에게 대학원 생활 전반에 대해 많은 조언과 도움을 받았습니다.

제가 deblurring 연구를 수행하는 데 도움을 주기 위해 수고를 아끼지 않았던 김태현 선배에게 감사합니다. 기존 연구를 어떻게 발전시켜 나아갈 지, 소스코드 관리는 어떻게 해야 하는지 등 연구하는 방법을 체득하는 데 큰 도움을 주었습니다.

덕분에 제가 이후로도 독자적인 연구를 수행할 수 있는 기반을 닦을 수 있었습니다. 안병주 선배 또한 김태현 선배와 함께 먼저 deblurring 연구를 수행한 선례를 보여주어 제가 같은 분야의 연구를 이어나가는 데 이정표가 되었습니다.

그리고 뿌듯한 연구 결과인 EDSR을 함께 만들었던 동료들인 비, 상현이, 희원이형에게 감사합니다. 개인적으로 EDSR을 통해 얻은 가장 큰 성과는 팀워크라고 생각합니다. 다함께 아이디어를 내고 역할을 나누어 실험하던 시간은 협력의 가치를 느낄 수 있었던 값진 경험이었고 이후로도 적극적으로 공동연구를 진행하는 원동력이 되었습니다.

또 REDS 데이터셋을 만들 때 데이터 촬영을 위해 힘써준 성용이, 석일이, 경식이, 상현이에게 감사합니다. 데이터 취득은 직접 다양한 장소에 가서 촬영을 해야 하는 등 다른 알고리즘 연구와 달리 수고롭고 번거로운 일입니다. 때문에 많은 사람들이 기피하는 일인데 기꺼이 함께 데이터 취득을 위해 많은 노력과 시간을 들여준 동료들입니다. 그리고 REDS 데이터르 기반으로 한 NTIRE 및 AIM 국제 워크샵 및 대회 운영에 함께 참여해준 상현이, 재린이, 수영이에게 감사합니다. 해마다 규모가 커지는 국제 행사 관리는 혼자 관리하기 어려운 일인데 좋은 동료들이 함께 부담을 나눠주는 도움을 받아 성공적으로 운영할 수 있었습니다.

그리고 박사과정 기간 중에 해외에 나가 Microsoft Research에서 Sing Bing Kang과, 그리고 Max-Planck Institute for Intelligent Systems에서 Andreas Geiger와 함께 일할 수 있는 기회를 가질 수 있었습니다. 대학원 생활동안 학교 뿐만 아니라 외부 기관에서 서로 다른 방식의 연구와 조직문화를 체험해볼 수 있는 기회였습니다. 더불어 ETH Zürich의 Radu Timofte와 함께 NTIRE 및 AIM 워크샵, 챌린지를 공동 운영하며 협력할 수 있었던 것 또한 개인으로서 쉽게 얻기 힘든 경험이었습니다.

제가 MSR 인턴 활동을 위해 미국에 체류할 때 반겨주시고 적응을 위해 도와주었던 선배들께 감사합니다. 유학생활 중이던 노준하 선배는 제가 미국에 처음 도착했을 때 집에 초대해 주시며 현지 생활에 필요한 정보를 알려주셨습니다. 김재철, 최종현 선배는 미국 회사 경험을 알려주시면서 회사 생활과 진로에 대한 조언을 해주셨습니다. 또 우리 연구실에서 석사학위를 받고 독일 MPI-IS에 박사과정 유학 중이던 Kamil은 제가 방문연구원으로 독일에 체류할 때 적응할 수 있도록 도와주고

여러 사람들을 소개해 주었습니다. 또 김효진, 권정현 선배도 학회에서 만났을 때를 포함하여 종종 해외 회사 생활 및 진로에 대해 깊게 상담해 주었습니다.

그리고 연구실 생활을 하면서 좋은 동료이자 친구가 되어준 선후배들에게 감사합니다. 제가 신입생일 때 신입생 교육을 해주시던 때부터 언제나 따뜻하고 친절한 모습을 보여준 해솔이형에게 감사합니다. 배려심, 멋짐과 함께 이성적인 면모와 판단력도 함께 가진 해솔이형은 대학원 생활 전반에 걸쳐 든든한 버팀목이 되어주었습니다. 희수 형은 연구에 대한 열정과 헌신을 보여주어 연구 동기에 대한 자극을 받을 수 있었습니다. 제가 신입생으로 처음 만났을 때 희수 형이 막 해외 인턴을 마치고 돌아온 날부터 바로 밤새워 연구하던 모습이 기억에 남습니다. 유민누나와 승연이형도 오랜 시간동안 연구실 생활을 함께하면서 연구 분야는 달라도 서로의 생각을 교환하고 토론하는 시간을 보냈습니다. 연구에 매진하다 보면 자기 분야에만 몰두하기 쉬운데 서로 다른 분야를 함께 이해하는 데 많은 도움을 받았습니다. 광모 형은 인간적인 여유과 놀라운 집중력을 동시에 보여주었고 이를 통해 연구에 성실히 임하면서도 지치지 않을 수 있도록 지속가능한 생활을 유지하고 있는지 스스로를 되돌아볼 수 있었습니다. 장훈이형에게서는 끊임없는 노력을 통해 새로운 트렌드를 파악하고 비판적인 시각을 견지하는 모습을 볼 수 있었습니다. 변화가 빠르고 새로운 연구가 점점 많아지는 컴퓨터 비전 분야 특성상 연구자로서 갖춰야 할 중요한 소양을 배울 수 있어 감사합니다.

그리고 언제나 따뜻한 인간미를 보여준 의영이형에게 감사합니다. 지홍이형도 어른스러움으로 동생들을 이끌어주었고 여러 좋은 기회와 연구 환경을 만들기 위해 노력하는 모습을 보고 연구 외적인 노력도 중요하다는 것을 깨달을 수 있었습니다. 학부 시절부터 같은 반 동기였고 지금까지 연구실 사람들 중 가장 오래 알고 지낸 친구 동우에게도 감사합니다. 그리고 경식이도 오랜 시간 동안 열정과 남다른 패기를 가지고 치밀하게 연구하는 모습이 제게 큰 귀감이 되었습니다. 성용이는 연구실에 밝은 분위기를 가져올 뿐만 아니라 연구실 사람들과 함께 성장하며 연구를 수행해 온 훌륭한 친구이자 동료로서 모두에게 긍정적으로 기여하였습니다. 비에게도 많은 고마움을 느낍니다. NTIRE 챌린지에 함께 참가하자고 먼저 제안해주었습니다. 우리 팀의 NTIRE 챌린지 우승을 통해 많은 후속 연구가 이루어질 수 있었고

귀한 계기를 만들어 주어 고맙습니다. 그 과정에서 비는 스스로 적극적으로 연구를 이끌어가며, 분산된 아이디어들을 통합하는 등 제가 함께하면서 보고 배울 점이 많았습니다. 준형이도 집중적으로 super-resolution 연구에 임하여 국제 대회에서 좋은 성적을 거두었고 제게 좋은 자극을 주었습니다.

그리고 학부 졸업프로젝트 조교와 학생으로 처음 만난 이후로 지금까지 가장 많이 연구를 함께한 상현이에게 깊은 감사를 전합니다. 아이디어가 생겼을 때 주저없이 논의할 수 있고 그렇게 만든 아이디어로 연구를 수행해나가는 과정을 함께 겪을 수 있는 동료를 가졌다는 것이 제 박사과정에서의 큰 행운이라고 생각합니다. 또한 상현이는 번뜩이는 아이디어와 빠른 수행능력으로 연구를 보다 수월하게 만들 뿐만 아니라 새로운 기법을 도입하는데 주저하지 않아 저를 여러 번 기쁜 마음으로 놀라게 해 주었습니다. 지금까지 지켜본 상현이는 개인으로서도 매우 뛰어날 뿐만 아니라 다른 사람에게 도움을 주는 데에도 능숙한 친구입니다. 지금까지 상현이와 함께한 모든 경험이 자랑스럽습니다.

그리고 희원이형에게도 함께 일하면서 연구적으로 뿐만 아니라 인격, 끝없는 인내심과 굳건한 의지를 보고 배울 수 있었습니다. 함께하는 일들에서 언제나 희원이형이 깊은 생각을 하고 있는 것을 느낄 수 있었고 언제나 유머를 잃지 않는 모습에서 정신력과 삶의 지혜를 느낄 수 있었습니다.

멀리 타향에서 와 연구실 생활을 하고 다시 박사 유학의 길을 떠난 Amin도 그동안 보여준 배려심과 성실함에 감사합니다. Mohsen과 Reyhaneh도 타향에까지 찾아와 연구에 임하는 적극적인 모습을 보여주었습니다. 함께 과제를 수행하면서 고생한 건운이, 발표자료 준비 및 여러 행사에서 미적 감각을 발휘해준 석일이에게도 감사합니다.

수영이, 재영이, 재하, 강건이, 영욱이는 많은 사람들을 위해 연구실에 봉사해 주었습니다. 수영이와 재영이는 본인 연구를 수행할 뿐만 아니라 저를 포함한 다른 사람들이 연구에 집중할 수 있도록 서버 관리를 맡아 많은 시간과 수고를 들였습니다. 또한 서버 구매에도 많은 노동이 필요한데 재하와 강건이가 함께 많이 수고해 주었습니다. 그리고 대학원 생활을 하다 보면 의식적으로 노력하지 않으면 건강을 돌아보지 않게 되는 경우가 많은데 영욱이는 연구와 동시에 연구실 건강 증진을 위

해 많이 수고해 주었습니다.

그리고 고맙게도 많은 후배들이 deblur 연구에 관심을 가지고 공동연구에 참여해 주었습니다. 재하는 인턴으로 처음 만났을 때부터 좋은 연구능력을 발휘해 주었고 좋은 아이디어가 있을 때 함께 연구를 하자고 제안해주었으며 적극적인 의지를 보여주고 있습니다. 재린이는 연구에 대한 대단한 열의로 저에게 자극을 주었을 뿐만 아니라 제 연구에도 함께 참여하여 실험 결과 해석 등 많은 도움을 주었습니다. 또한 군대에 가서까지도 연구를 놓지 않는 등 연구에 대한 남다른 열정을 보여주었습니다. 준규도 연구를 시작한 지 얼마 되지 않아서부터 여러 흥미로운 아이디어를 내고 능동적인 태도로 실험을 주도하여 저를 여러 차례 놀라게 해 주었습니다. 디블러링 연산 가속화 연구를 위해 함께 노력을 기울이고 있는 지훈이, 정훈이, 채은이에게도 감사합니다. 후속 데이터 작업을 위해 관심을 가지고 있고 종종 아이디어를 함께 논의한 현진이에게도 감사합니다. 이렇게 함께 할 훌륭한 동료가 많이 있는 환경에서 연구할 수 있었다는 것은 쉽게 가질 수 없는 복이라 생각하며 모든 동료들에게 감사합니다.

또한 직접적으로 같은 프로젝트에 참여하지 않더라도 곁에서 의욕적인 태도와 굳건한 의지를 놓지 않는 모습을 보여주는 홍석이에게도 감사합니다. 비슷한 분야를 연구하는 우석이, 컴퓨터 비전 연구실의 새로운 미래를 책임질 도희, 형진이, 상민이, 서연님 모두에게도 큰 기대와 함께 감사를 전합니다.

마지막으로 언제나 곁에서 든든하게 지지해주고 지원을 아끼지 않으신 부모님께 감사드립니다.