

Scale-recurrent Network for Deep Image Deblurring

Xin Tao^{1,2,*} Hongyun Gao^{1,2,*} Xiaoyong Shen² Jue Wang³ Jiaya Jia^{1,2}

¹The Chinese University of Hong Kong ²YouTu Lab, Tencent ³Megvii Inc.

{xtao, hygao}@cse.cuhk.edu.hk {goodshenxy, arphid, leojia9}@gmail.com

Abstract

In single image deblurring, the “coarse-to-fine” scheme, i.e. gradually restoring the sharp image on different resolutions in a pyramid, is very successful in both traditional optimization-based methods and recent neural-network-based approaches. In this paper, we investigate this strategy and propose a Scale-recurrent Network (SRN-DeblurNet) for this deblurring task. Compared with the many recent learning-based approaches in [25], it has a simpler network structure, a smaller number of parameters and is easier to train. We evaluate our method on large-scale deblurring datasets with complex motion. Results show that our method can produce better quality results than state-of-the-arts, both quantitatively and qualitatively.

1. Introduction

Image deblurring has long been an important task in computer vision and image processing. Given a motion- or focal-blurred image, caused by camera shake, object motion or out-of-focus, the goal of deblurring is to recover a sharp latent image with necessary edge structures and details.

Single image deblurring is highly ill-posed. Traditional methods applied various constraints to model characteristics of blur (e.g. uniform/non-uniform/depth-aware), and utilized different natural image priors [1, 3, 6, 39, 14, 40, 26, 27] to regularize the solution space. Most of these methods involve heuristic parameter-tuning and expensive computation. Further, the simplified assumptions on the blur model often hinder their performance on real-word examples, where blur is far more complex than modeled and is entangled with in-camera image processing pipeline.

Learning-based methods have also been proposed for deblurring. Early methods [29, 34, 38] substitute a few modules or steps in traditional frameworks with learned parameters to make use of external data. More recent work started to use end-to-end trainable networks for deblurring images [25] and videos [18, 33, 37]. Among them, Nah *et*

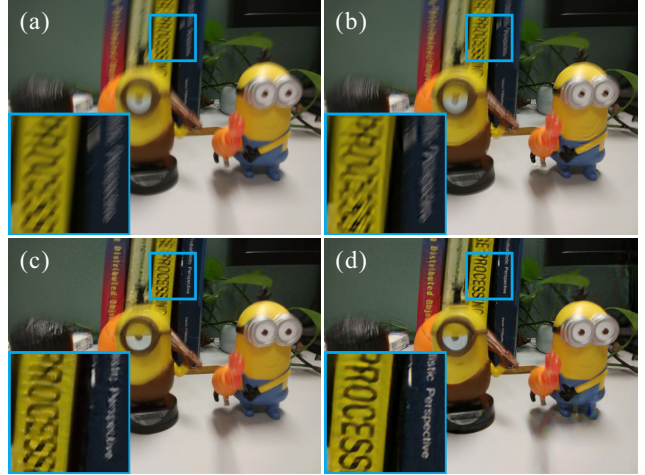


Figure 1. **One real example.** (a) Input blurred image. (b) Result of Sun *et al.* [34]. (c) Result of Nah *et al.* [25]. (d) Our result.

al.[25] have achieved state-of-the-art results using a multi-scale convolutional neural network (CNN). This method commences from a very coarse scale of the blurry image, and progressively recovers the latent image at higher resolutions until the full resolution is reached. This framework follows the multi-scale mechanism in traditional methods, where the coarse-to-fine pipelines are common when handling large blur kernels [6].

In this paper, we explore a more effective network structure for multi-scale image deblurring. We propose the new scale-recurrent network (SRN), which addresses two important and general issues in CNN-based deblurring systems.

Scale-recurrent Structure In well-established multi-scale methods, the solver and corresponding parameters at each scale are usually the same. This is intuitively a natural choice since in each scale we aim to solve the same problem. It was also found that varying parameters at each scale could introduce instability and cause the extra problems of unrestrictive solution space. Another concern is that input images may have different resolutions and motion scales. If parameter tweaking in each scale is allowed, the solution

* These two authors contributed equally to this work.

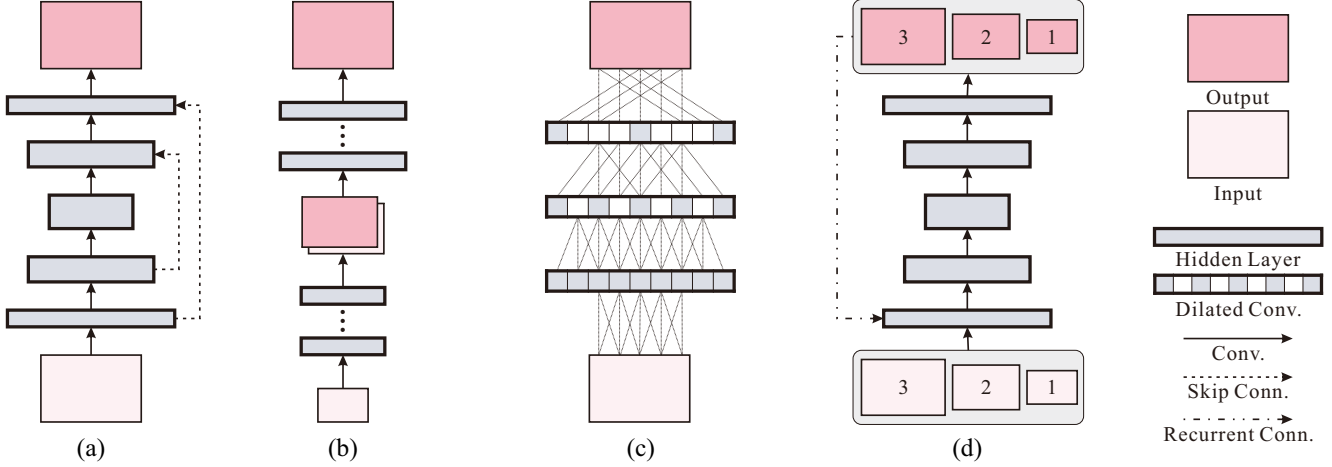


Figure 2. **Different CNNs for image processing.** (a) U-Net [28] or encoder-decoder network [24]. (b) Multi-scale [25] or cascaded refinement network [4]. (c) Dilated convolutional network [5]. (d) Our proposed scale-recurrent network (SRN).

may overfit to a specific image resolution or motion scale.

We believe this scheme should also be applied to CNN-based methods for the same reasons. However, recent cascaded networks [4, 25] still use independent parameters for each of their scales. In this work, we propose sharing network weights across scales to significantly reduce training difficulty and introduce obvious stability benefits.

The advantage is twofold. First, it significantly reduces the number of trainable parameters. Even with the same training data, the recurrent exploitation of shared weights works in a way similar to using data multiple times to learn parameters, which actually amounts to data augmentation regarding scales. Second, our proposed structure can incorporate recurrent modules, where the hidden state captures useful information and benefits restoration across scales.

Encoder-decoder ResBlock Network Also inspired by recent success of encoder-decoder structure for various computer vision tasks [23, 33, 35, 41], we explore the effective way to adapt it in image deblurring. In this paper, we show that directly applying existing encoder-decoder structure cannot produce optimal results. Our Encoder-decoder ResBlock network, on the contrary, amplifies the merit of various CNN structures and yields the feasibility in training. It also produces a very large receptive field, which is of vital importance for large-motion deblurring.

Our experiments show that with the recurrent structure and combining above advantages, our end-to-end deep image deblurring framework can greatly improve training efficiency ($\approx 1/4$ training time of [25] to accomplish similar restoration). We only use less than $1/3$ trainable parameters with much faster testing time. Besides training efficiency, our method can also produce higher quality results than existing methods both quantitatively and qualitatively,

as shown in Fig. 1 and to be elaborated later. We name this framework *scale-recurrent network* (SRN).

2. Related Work

In this section, we briefly review image deblurring methods and recent CNN structures for image processing.

Image/Video Deblurring After the seminal work of Fergus *et al.* [12] and Shan *et al.* [30], many deblurring methods were proposed towards both restoration quality and adaptiveness to different situations. Natural image priors were designed to suppress artifacts and improve quality. They include total variation (TV) [3], sparse image priors [22], heavy-tailed gradient prior [30], hyper-Laplacian prior [21], l_0 -norm gradient prior [40], *etc.* Most of these traditional methods follow the coarse-to-fine framework. Exceptions include frequency-domain methods [8, 14], which are only applicable to limited situations.

Image deblurring also benefits from recent advancement of deep CNN. Sun *et al.* [34] used the network to predict blur direction and width. Schuler *et al.* [29] stacked multiple CNNs in a coarse-to-fine manner to simulate iterative optimization. Chakrabarti [2] predicted deconvolution kernel in frequency domain. These methods follow the traditional framework with several parts replaced by CNN modules. Su *et al.* [33] used an encoder-decoder network with skip-connections to learn video deblurring. Nah *et al.* [25] trained a multi-scale deep network to progressively restore sharp images. These end-to-end methods make use of multi-scale information via different structures.

CNNs for Image Processing Different from classification tasks, networks for image processing require special design. As one of the earliest methods, SRCNN [9] used 3

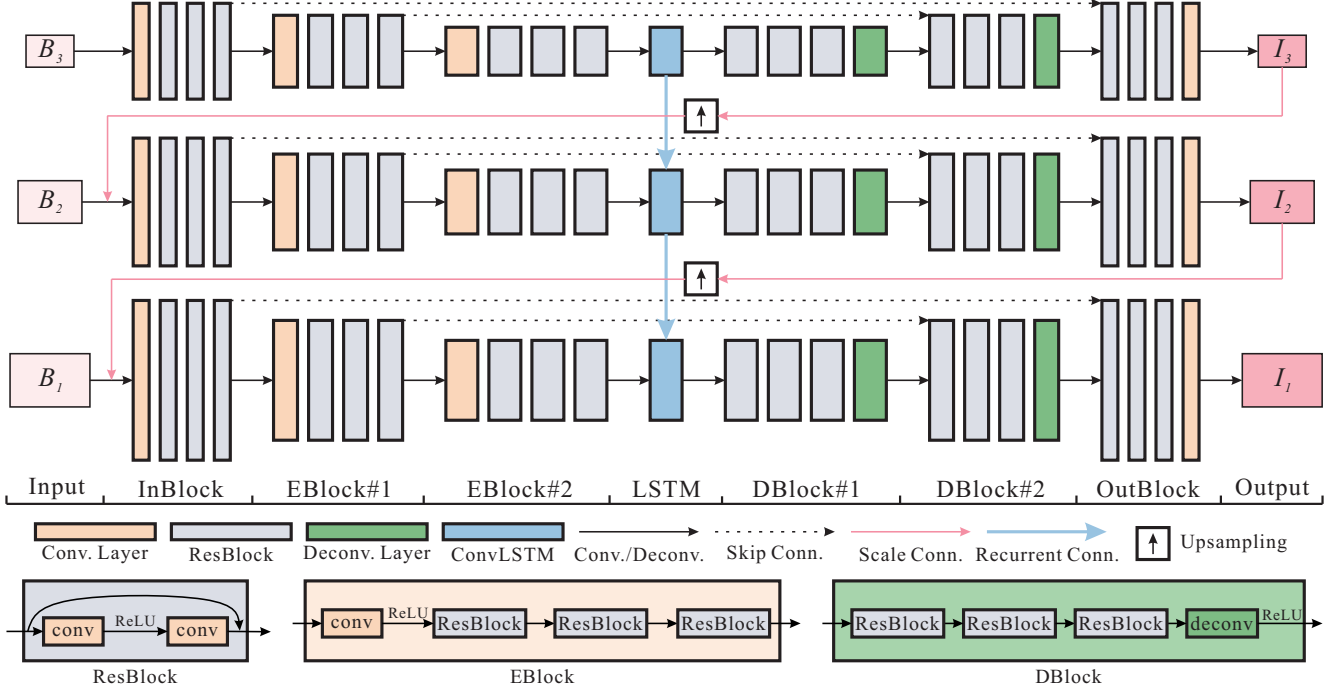


Figure 3. Our proposed SRN-DeblurNet framework.

flat convolution layers (with the same feature map size) for super-resolution. Improvement was yielded by U-net [28] (as shown in Fig. 2(a)), also termed as encoder-decoder networks [24], which greatly increases regression ability and is widely used in recent work of FlowNet [10], video deblurring [33], video super-resolution [35], frame synthesis [23], *etc.* Multi-scale CNN [25] and cascaded refinement network (CRN) [4] (Fig. 2(b)) simplified training by progressively refining output starting from a very small scale. They are successful in image deblurring and image synthesis, respectively. Fig. 2(c) shows a different structure [5] that used dilated convolution layers with increasing rates, which approximates increasing kernel sizes.

3. Network Architecture

The overall architecture of the proposed network, which we call SRN-DeblurNet, is illustrated in Fig. 3. It takes as input a sequence of blurry images downsampled from the input image at different scales, and produces a set of corresponding sharp images. The sharp one at the full resolution is the final output.

3.1. Scale-recurrent Network (SRN)

As explained in Sec. 1, we adopt a novel recurrent structure across multiple scales in the coarse-to-fine strategy. We form the generation of a sharp latent image at each scale as a sub-problem of the image deblurring task, which takes a blurred image and an initial deblurring result (upsampled

from previous scale) as input, and estimates the sharp image at this scale as

$$\mathbf{I}^i, \mathbf{h}^i = \text{Net}_{SR}(\mathbf{B}^i, \mathbf{I}^{i+1\uparrow}, \mathbf{h}^{i+1\uparrow}; \theta_{SR}), \quad (1)$$

where i is the scale index, with $i = 1$ representing the finest scale. $\mathbf{B}^i$ and $\mathbf{I}^i$ are the blurry and estimated latent images at the i -th scale, respectively. Net_{SR} is our proposed scale-recurrent network with training parameters denoted as θ_{SR} . Since the network is recurrent, hidden state features $\mathbf{h}^i$ flow across scales. The hidden state captures image structures and kernel information from the previous coarser scales. $(\cdot)^\uparrow$ is the operator to adapt features or images from the $(i + 1)$ -th to i -th scale.

Eq. (1) gives a detailed definition of the network. In practice, there is enormous flexibility in network design. First, recurrent networks can take different forms, such as vanilla RNN, long-short term memory (LSTM) [16, 32] and gated recurrent unit (GRU) [7]. We choose ConvLSTM [32] since it performs better in our experiments. Analysis will be given in Sec. 4. Second, possible choices for operator $(\cdot)^\uparrow$ include deconvolution layer, sub-pixel convolution layer [31] and image resizing. We use bilinear interpolation for all our experiments for its sufficiency and simplicity. Third, the network at each scale needs to be properly designed for optimal effectiveness to recover the sharp image. Our method is detailed in the following.

3.2. Encoder-decoder with ResBlocks

Encoder-decoder Network Encoder-decoder network [24, 28] refers to the symmetric CNN structures that first progressively transform input data into feature maps with smaller spatial sizes and more channels (in encoder), and then transform them back to the shape of the input (in decoder). Skip-connections between corresponding feature maps are widely used to combine different levels of information. They can also benefit gradient propagation and accelerate convergence. Typically, the encoder contains several stages of convolution layers with strides, and the decoder module is implemented using a series of deconvolution layers [23, 33, 35] or resizing. Additional convolution layers are inserted after each level to further increase depth.

The encoder-decoder structure has been proven to be effective in many vision tasks [23, 33, 35, 41]. However, directly using the encoder-decoder network is not the best choice for our task with the following considerations.

First, for the task of deblurring, the receptive field needs to be large to handle severe motion, resulting in stacking more levels for encoder/decoder modules. However, this strategy is not recommended in practice since it increases the number of parameters quickly with the large number of intermediate feature channels. Besides, the spatial size of middle feature map would be too small to keep spatial information for reconstruction. Second, adding more convolution layers at each level of encoder/decoder modules would make the network slow to converge (with flat convolution at each level). Finally, our proposed structure requires recurrent modules with hidden states inside.

Encoder/decoder ResBlock We make several modifications to adapt encoder-decoder networks into our framework. First, we improve encoder/decoder modules by introducing residual learning blocks [15]. Based on results of [25] and our experiments, we choose to use ResBlocks instead of the original one in ResNet [15] (without batch normalization). As illustrated in Fig. 3, our proposed Encoder ResBlocks (EBlocks) contains one convolution layer followed by several ResBlocks. The stride for convolution layer is 2. It doubles the number of kernels of previous layer and downsamples the feature maps to half size. Each of the following ResBlocks contains 2 convolution layers. Besides, all convolution layers have the same number of kernels. Decoder ResBlock (DBlocks) is symmetric to EBlock. It contains several ResBlocks followed by one deconvolution layer. The deconvolution layer is used to double the spatial size of feature maps and halve channels.

Second, our scale-recurrent structure requires recurrent modules inside networks. Similar to the strategy of [35], we insert convolution layers in the bottleneck layer for hidden state to connect consecutive scales. Finally, we use large convolution kernels of size 5×5 for every convolution layer.

The modified network is expressed as

$$\begin{aligned} \mathbf{f}^i &= \text{Net}_E(\mathbf{B}^i, \mathbf{I}^{i+1\uparrow}; \theta_E), \\ \mathbf{h}^i, \mathbf{g}^i &= \text{ConvLSTM}(\mathbf{h}^{i+1\uparrow}, \mathbf{f}^i; \theta_{LSTM}), \\ \mathbf{I}^i &= \text{Net}_D(\mathbf{g}^i; \theta_D), \end{aligned} \quad (2)$$

where Net_E and Net_D are encoder and decoder CNNs with parameters θ_E and θ_D . 3 stages of EBlocks and DBlocks are used in Net_E and Net_D , respectively. θ_{LSTM} is the set of parameters in ConvLSTM. Hidden state h^i may contain useful information about intermediate result and blur patterns, which is passed to the next scale and benefits the fine-scale problem.

The details of model parameters are specified here. Our SRN contains 3 scales. The $(i+1)$ -th scale is of half size of the i -th scale. For the encoder/decoder ResBlock network, there are 1 InBlock, 2 EBlocks, followed by 1 Convolutional LSTM block, 2 DBlocks and 1 OutBlock, as shown in Fig. 3. InBlock produces a 32-channel feature map. OutBlock takes previous feature map as input and generates output image. The numbers of kernels of all convolution layers inside each EBlock/DBlock are the same. For EBlocks, the numbers of kernels are 64 and 128, respectively. For DBlocks, they are 128 and 64. The stride size for the convolution layer in EBlocks and deconvolution layers is 2, while all others are 1. Rectified Linear Units (ReLU) are used as the activation function for all layers, and all kernel sizes are set to 5.

3.3. Losses

We use Euclidean loss for each scale, between network output and the ground truth (downsampled to the same size using bilinear interpolation) as

$$\mathcal{L} = \sum_{i=1}^n \frac{\kappa_i}{N_i} \|\mathbf{I}^i - \mathbf{I}_*^i\|_2^2, \quad (3)$$

where $\mathbf{I}^i$ and $\mathbf{I}_*^i$ are our network output and ground truth respectively in the i -th scale. $\{\kappa_i\}$ are the weights for each scale. We empirically set $\kappa_i = 1.0$. N_i is the number of elements in $\mathbf{I}^i$ to normalize. We have also tried total variation and adversarial loss. But we notice that L_2 -norm is good enough to generate sharp and clear results.

4. Experiments

Our experiments are conducted on a PC with Intel Xeon E5 CPU and an NVIDIA Titan X GPU. We implement our framework on TensorFlow platform [11]. Our evaluation is comprehensive to verify different network structures, as well as various network parameters. For fairness, unless noted otherwise, all experiments are conducted on the same dataset with the same training configuration.

Data Preparation To create a large training dataset, early learning-based methods [2, 29, 34] synthesize blurred images by convolving sharp images with real or generated uniform/non-uniform blur kernels. Due to the simplified image formation models, the synthetic data is still different from those captured by cameras. Recently, researchers [25, 33] proposed generating blurred images through averaging consecutive short-exposure frames from videos captured by high-speed cameras, e.g. GoPro Hero 4 Black, to approximate long-exposure blurry frames. These generated frames are more realistic since they can simulate complex camera shake and object motion, which are common in real photographs.

For fair comparison with respect to the network structure, we train our network using the GOPRO dataset of [25], which contains 3,214 blurry/clear image pairs. Following the same strategy as in [25], we use 2,103 pairs for training and the remaining 1,111 pairs for evaluation.

Model Training For model training, we use Adam solver [19] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The learning rate is exponentially decayed from initial value of 0.0001 to $1e^{-6}$ at 2000 epochs using power 0.3. According to our experiments, 2,000 epochs are enough for convergence, which takes about 72 hours. In each iteration, we sample a batch of 16 blurry images and randomly crop 256×256 -pixel patches as training input. Ground truth sharp patches are generated accordingly. All trainable variables are initialized using Xavier method [13]. The parameters described above are fixed for all experiments.

For experiments that involve recurrent modules, we apply gradient clip only to weights of ConvLSTM module (clipped by global norm 3) to stabilize training. Since our network is fully convolutional, images of arbitrary size can be fed in it as input, as long as GPU memory allows. For a testing image of size 720×1280 , running time of our proposed method is around 1.87 seconds.

Table 1. Quantitative results of the baseline models.

Model	SS	SC	w/o R	RNN	SR-Flat
Param	2.73M	8.19M	2.73M	3.03M	2.66M
PSNR	28.40	29.05	29.26	29.35	27.53
SSIM	0.9045	0.9166	0.9197	0.9210	0.8886
Model	SR-RB	SR-ED	SR-EDRB1	SR-EDRB2	SR-EDRB3
Param	2.66M	3.76M	2.21M	2.99M	3.76M
PSNR	28.11	29.06	28.60	29.32	29.98
SSIM	0.8991	0.9170	0.9082	0.9204	0.9254

4.1. Multi-scale Strategy

To evaluate the proposed scale-recurrent network, we design several baseline models. To evaluate network structures, we use kernel size 3 for all convolution layers for reasonable efficiency. Single-scale model **SS** uses the same

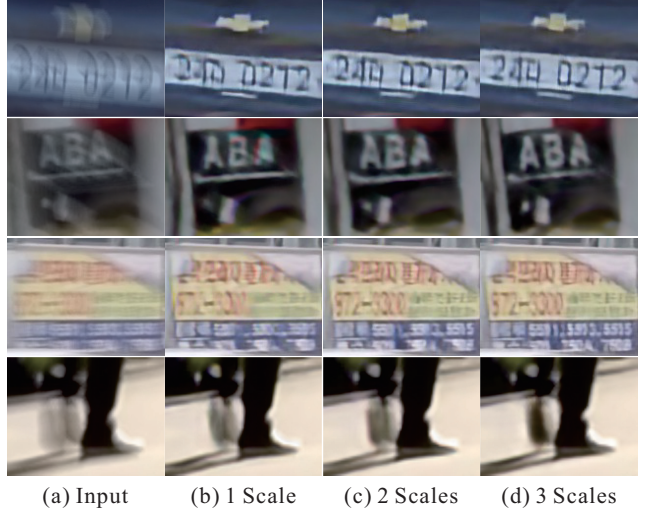


Figure 4. Results of the multi-scale baseline method.

structure as our proposed one, except that only a single-scale image is taken as input at its original resolution. Recurrent modules are replaced by one convolution layer to ensure the same number of convolution layers.

Baseline model **SC** refers to the scale-cascaded structure as in [4, 25], which uses 3 stages of independent networks. Each single-stage network is the same as model **SS**. Therefore, the trainable parameters of this model are 3 times more than our method. Model **w/oR** does not contain explicit recurrent modules in bottleneck layer (*i.e.* model **SS**), which is a shared-weight version of model **SC**. Model **RNN** uses vanilla RNN structure instead of ConvLSTM.

The results of different methods on the testing dataset are shown in Table 1, from which we make several useful observations. First, the multi-scale strategy is very effective for the image deblurring task. Model **SS** uses the same structure and the same number of parameters as our proposed SRN structure, and yet performs much worse in terms of PSNR (28.40dB vs. 29.98dB). One visual comparison is given in Fig. 4 where the single-scale Model **SS** in (b) can recover structure from severely blurred images. But the characters are still not clear enough for recognition.

Results are improved when we use 2 scales as shown in Fig. 4(c), because multi-scale information has been effectively incorporated. The more complete model with 3 scales further produces better results in Fig. 4(d); but the improvement is already minor.

Second, independent parameters for each scale are *not* necessary and may be even harmful, proved by the fact that Model **SC** performs worse than Model **w/oR**, **RNN** and **SR-EDRB3** (which share the same Encoder-decoder ResBlock structure with 3 ResBlocks). We believe the reason is that, although more parameters lead to a larger model, it

also requires longer training time and larger training dataset. In our constrained setting with fixed dataset and training epochs, model **SC** may not be optimally trained.

Finally, we also test different recurrent modules. The results show that vanilla RNN is better than not using RNN, and ConvLSTM achieves the best results with model **SR-EDRB3**.

4.2. Encoder-decoder ResBlock Network

We also design a series of baseline models to evaluate the effectiveness of the encoder-decoder with ResBlock structure. For fair comparison, all models here use our scale-recurrent (**SR**) framework. Model **SR-Flat** replaces encoder-decoder architecture with flat convolution layers, the number of which is the same as that of the proposed network, *i.e.* 43 layers. Model **SR-RB** replaces all EBlocks and DBlocks with ResBlock. No stride or pooling is included. This makes feature maps have the same size. Model **SR-ED** uses original encoder-decoder structure, with all ResBlocks replaced with 2 convolution layers. We also compare with different numbers of ResBlocks in EBlock/DBlock. Models **SR-EDRB1**, **SR-EDRB2** and **SR-EDRB3** refer to models using 1, 2 and 3 ResBlocks, respectively.

Quantitative results are shown in Table 1. Flat convolution model **Flat** performs worst in terms of both PSNR and SSIM. In our experiments, it takes significantly more time to reach the same level of quality as other results. Model **RB** is much better, since ResBlock structure is designed for better training. The best results are accomplished by our proposed model **SR-EDRB1-3**. The quantitative results also get better as the number of ResBlocks increases. We choose 3 ResBlocks in our proposed model, since the improvement beyond 3 ResBlocks is marginal and it is a good balance between efficiency and performance.

4.3. Comparisons

We compare our method with previous state-of-the-art image deblurring approaches on both evaluation datasets and real images. Since our model deals with general camera shake and object motion (*i.e.* dynamic deblurring [17]), it is unfair to compare with traditional uniform deblurring methods. The method of Whyte *et al.* [36] is selected as a representative traditional method for non-uniform blur. Note that for most examples in the testing dataset, blurred images are caused merely by camera shake. Thus the non-uniform assumption in [36] holds.

The method of Kim *et al.* [17] should be able to handle dynamic blurring. But no code or executable is provided. Instead we compare with more recent work of Nah *et al.* [25], which demonstrated very good results. Sun *et al.* [34] estimated blur kernels using CNN, and used traditional deconvolution methods to recover the sharp image. We use official implementation from the authors with de-

Table 2. Quantitative results on “test” dataset (in terms of PSNR/SSIM).

Method	GOPRO		Köhler Dataset		Time
	PSNR	SSIM	PSNR	MSSIM	
Kim <i>et al.</i>	23.64	0.8239	24.68	0.7937	1 hr
Sun <i>et al.</i>	24.64	0.8429	25.22	0.7735	20 min
Nah <i>et al.</i>	29.08	0.9135	26.48	0.8079	3.09 s
Ours	30.26	0.9342	26.75	0.8370	1.87s

fault parameters. The quantitative results on GOPRO testing set and Köhler Dataset [20] are listed in Table 2. Visual comparison is shown in Figs. 5 and 6. More results are included in our supplementary material.

Benchmark Datasets The first row of Fig. 5 contains images from the GOPRO testing datasets, which suffer from complex blur due to large camera and object motion. Although traditional method [36] models a general non-uniform blur for camera translation and rotation, it still fails for Fig. 5(a), (c), and (d), where camera motion dominates. It is because forward/backward motion, as well as scene depth, plays important roles in real blurred images. Moreover, violation of the assumed model results in annoying ringing artifacts, which make restored image even worse than input.

Sun *et al.* used CNN to predict kernel direction. But on this dataset, the complex blur patterns are quite different from their synthetic training set. Thus this method failed to predict reliable kernels on most cases, and results are only slightly sharpened. Recent state-of-the-art method [25] can produce good quality results, with remaining a few blurry structure and artifacts. Thanks to the designed framework, our method effectively produces superior results with sharper structures and clear details. According to our experiments, even on extreme cases, where motion is too large for previous solutions, our method can still produce reasonable results for important part and does not cause many visual artifacts on other regions, as shown in the last case of Fig. 6. Quantitative results are in accordance with our observation, where our framework outperforms others by a large margin.

Real Blurred Images The GoPro testing images are synthesized from high-speed cameras, which may still differ from real blurred input. We show our results on real-captured blurred images in Fig. 6. Our trained model generalizes well on these images, as shown in Fig. 6(d). Compared with results of Sun *et al.* and Nah *et al.*, ours are of high quality.

5. Conclusion

In this paper, we have explained what is the proper network structure for using the “coarse-to-fine” scheme in image deblurring. We have also proposed a scale-recurrent

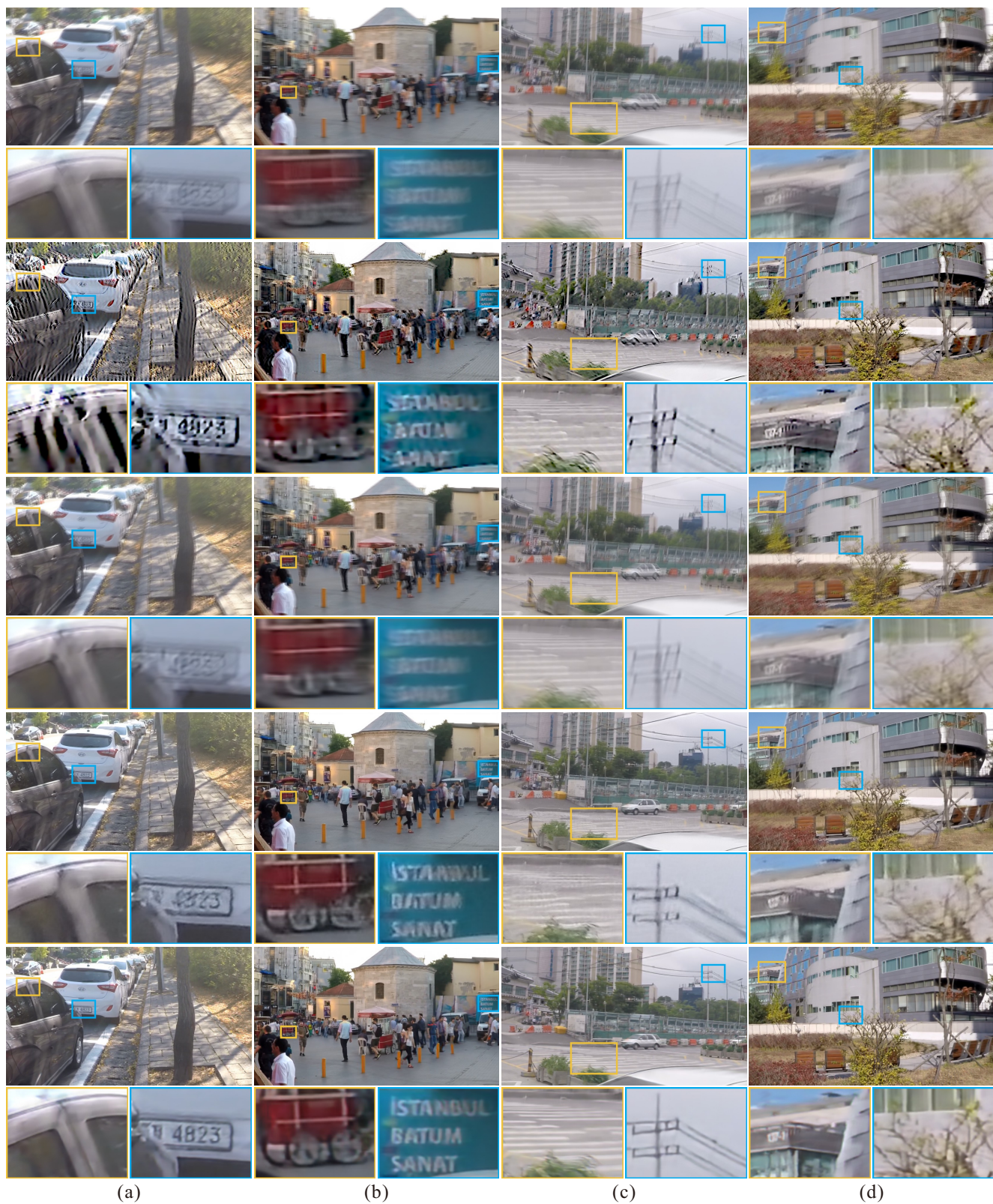
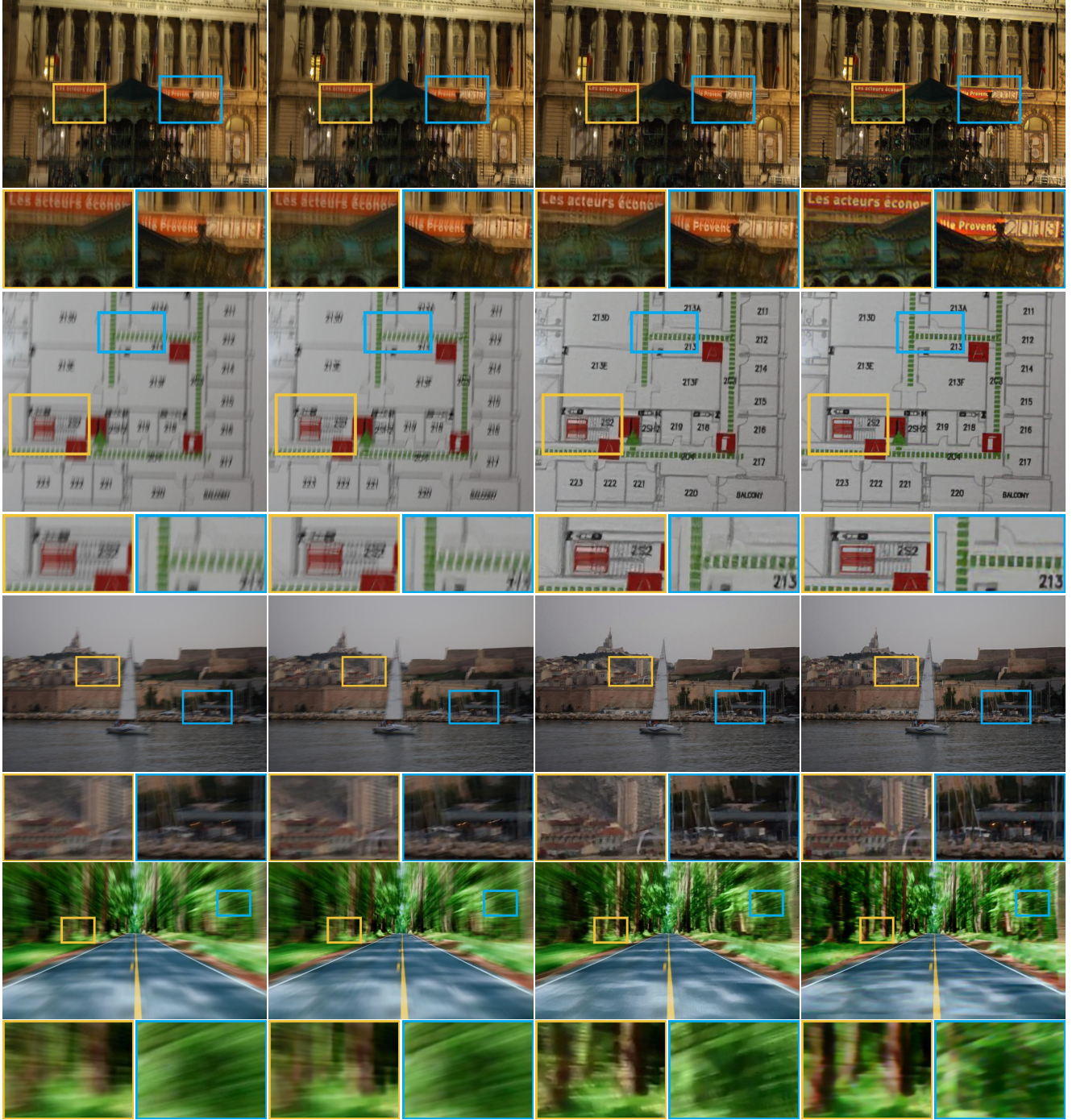


Figure 5. **Visual comparisons on testing dataset.** In the top-down order, we show input, results of Whyte *et al.* [36], Sun *et al.* [34], and Nah *et al.* [25], and our results (best view in high resolutions).



(a) Input

(b) Sun et al.

(c) Nah et al.

(d) Ours

Figure 6. Real-world blurred images.

network, as well as an encoder-decoder ResBlocks structure in each scale. This new network structure has less parameters than previous multi-scale deblurring ones and is easier to train. The results generated by our method are state-of-the-art, both qualitatively and quantitatively. We believe this scale-recurrent network can be applied to other image

processing tasks, and we will explore them in the future.

Acknowledgments

We thank our colleague Yi Wang for taking valuable discussion and Dr. Yuwing Tai for his generous help.

References

- [1] Y. Bahat, N. Efrat, and M. Irani. Non-uniform blind deblurring by reblurring. In *ICCV*, pages 3286–3294. IEEE, 2017.
- [2] A. Chakrabarti. A neural approach to blind motion deblurring. In *ECCV*, pages 221–235. Springer, 2016.
- [3] T. F. Chan and C.-K. Wong. Total variation blind deconvolution. *IEEE Trans. on Image Processing*, 7(3):370–375, 1998.
- [4] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*. IEEE, 2017.
- [5] Q. Chen, J. Xu, and V. Koltun. Fast image processing with fully-convolutional networks. In *ICCV*. IEEE, 2017.
- [6] S. Cho and S. Lee. Fast motion deblurring. In *ACM Trans. on Graphics*, volume 28, page 145. ACM, 2009.
- [7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [8] M. Delbracio and G. Sapiro. Burst deblurring: Removing camera shake through fourier burst accumulation. In *CVPR*, pages 2385–2393. IEEE, 2015.
- [9] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199. Springer, 2014.
- [10] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766. IEEE, 2015.
- [11] M. A. et. al. TensorFlow: large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [12] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. In *ACM Trans. on Graphics*, volume 25, pages 787–794. ACM, 2006.
- [13] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010.
- [14] A. Goldstein and R. Fattal. Blur-kernel estimation from spectral irregularities. In *ECCV*, pages 622–635. Springer, 2012.
- [15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE, 2016.
- [16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [17] T. Hyun Kim, B. Ahn, and K. Mu Lee. Dynamic scene deblurring. In *ICCV*, pages 3160–3167. IEEE, 2013.
- [18] T. Hyun Kim, K. Mu Lee, B. Scholkopf, and M. Hirsch. Online video deblurring via dynamic temporal blending network. In *ICCV*, pages 4038–4047. IEEE, 2017.
- [19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.
- [20] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf, and S. Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. pages 27–40, 2012.
- [21] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *NIPS*, pages 1033–1041, 2009.
- [22] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *CVPR*, pages 1964–1971. IEEE, 2009.
- [23] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *ICCV*. IEEE, 2017.
- [24] X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *NIPS*, pages 2802–2810, 2016.
- [25] S. Nah, T. H. Kim, and K. M. Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. pages 3883–3891, 2017.
- [26] J. Pan, Z. Hu, Z. Su, and M.-H. Yang. Deblurring text images via l0-regularized intensity and gradient prior. In *CVPR*, pages 2901–2908. IEEE, 2014.
- [27] J. Pan, D. Sun, H. Pfister, and M.-H. Yang. Blind image deblurring using dark channel prior. In *CVPR*, pages 1628–1636. IEEE, 2016.
- [28] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.
- [29] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. Learning to deblur. *TPAMI*, 38(7):1439–1451, 2016.
- [30] Q. Shan, J. Jia, and A. Agarwala. High-quality motion deblurring from a single image. volume 27, page 73. ACM, 2008.
- [31] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, pages 1874–1883. IEEE, 2016.
- [32] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, pages 802–810, 2015.
- [33] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang. Deep video deblurring. pages 1279–1288, 2017.
- [34] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *CVPR*, pages 769–777. IEEE, 2015.
- [35] X. Tao, H. Gao, R. Liao, J. Wang, and J. Jia. Detail-revealing deep video super-resolution. In *ICCV*. IEEE, 2017.
- [36] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. *International Journal on Computer Vision*, 98(2):168–186, 2012.
- [37] P. Wieschollek, M. Hirsch, B. Schölkopf, and H. P. Lensch. Learning blind motion deblurring. In *ICCV*. IEEE, 2017.
- [38] L. Xiao, J. Wang, W. Heidrich, and M. Hirsch. Learning high-order filters for efficient blind deconvolution of document photographs. In *ECCV*, pages 734–749. Springer, 2016.
- [39] L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. In *ECCV*, pages 157–170. Springer, 2010.
- [40] L. Xu, S. Zheng, and J. Jia. Unnatural l0 sparse representation for natural image deblurring. In *CVPR*, pages 1107–1114. IEEE, 2013.
- [41] N. Xu, B. Price, S. Cohen, and T. Huang. Deep image matting. In *CVPR*. IEEE, 2017.