Learning Spatially-Variant MAP Models for Non-blind Image Deblurring
– Supplemental Material –

Jiangxin Dong
MPI Informatics
Saarland Informatics Campus

Stefan Roth
TU Darmstadt
& hessian.AI

Bernt Schiele
MPI Informatics
Saarland Informatics Campus

Overview

In this supplemental material, we first present the details of the network architecture for predicting the spatially-variant filters in our SVMAP model, see Sec. A. Section B provides additional analyses of the proposed approach. Section C shows more visual comparisons against previous methods.

A. Configurations of the Spatially-Variant Filter Prediction Network

As discussed in the main paper, we use two deep neural networks to predict the spatially-variant filters, one for the data term and one for the regularization term. Table 5 lists the detailed configuration. Note that the networks for predicting the filters for the data and regularization terms share the same architecture, but the parameters are not shared.

Table 5. *Parameters of the spatially-variant filter prediction network. Conv* denotes a convolutional layer. *CR* denotes a convolutional layer followed by a ReLU. Recall that the filters for the data term and regularization term have $s_f \times s_f$ and $s_g \times s_g$ pixels, respectively. The number of filters for the data term and regularization term are M and N, respectively. The number of output features, n_f, is thus set as $s_f^2 M$ when predicting the filters for the data term and $s_g^2 N$ for the regularization term.

Layers	Kernel size	Number of features	Stride
CR_1	3×3	64	1
CR_2	3×3	64	1
CR_3	3×3	64	1
CR_4	3×3	64	1
CR_5	3×3	64	1
Conv	3×3	n_f	1

B. Additional Analyses and Discussions

B.1. Spatially-variant *vs.* invariant MAP models

Our goal in this section is to complement the discussion on the *effect of predicting spatially-variant filters* in Sec. 5 of the main paper. In order to demonstrate the effectiveness of our learned spatially-variant MAP model (SVMAP), we compare to a learned spatially-invariant model (*SIMAP* for short) in Tab. 4 of the main paper. For this baseline, since the pixel-dependent weights $\{\omega_i^d, \omega_j^r\}$ in Eq. (11) of the main paper cannot be absorbed into the spatially-uniform filters $\{f_i, g_j\}$, we need to predict $\{\omega_i^d, \omega_j^r, f_i, g_j\}$ by appropriate networks. For fair comparison, we use the same network architecture as for our approach (Tab. 5) to predict the spatially-uniform filters $\{f_i, g_j\}$. The network outputs $s_f^2 M$ and $s_g^2 N$ features for predicting $\{f_i\}$ and $\{g_j\}$, respectively, in feature maps that have the same size as the input image. To obtain the spatially-uniform filters, we apply global average pooling to generate $s_f^2 M$ and $s_g^2 N$ features of 1×1 pixels. Then we reshape the outputs to M filters of $s_f \times s_f$ pixels to obtain $\{f_i\}$, and to N filters of $s_g \times s_g$ pixels to yield $\{g_j\}$. The network parameters are not shared, as in the spatially-variant case.

Table 6. *Parameters of the pixel-dependent weight prediction network for the regularization term in the SIMAP baseline. CR* denotes a convolutional layer followed by a ReLU. We denote the number of filters for the regularization term as N.

Layers	Kernel size	Number of features	Stride
CR_1	3×3	64	1
CR_2	3×3	64	1
CR_3	3×3	64	1
CR_4	3×3	64	1
CR_5	3×3	64	1
CR_6	3×3	N	1

The detailed network architectures for predicting $\{\omega_j^r\}$ and $\{\omega_i^d\}$ are shown in Tabs. 6 and 7. As discussed in previous work [3, 6, 27], the robust penalty function for the data term should be less sensitive to outlying measurements. Note that the weights $\{\omega_i^d\}$ in the IRLS iteration are proportional to the derivative of the penalty function. When the value of the residual image increases beyond a threshold, these weights should not increase further, as these locations likely correspond to outliers. Thus, we adopt a sigmoid output layer to regularize the network for predicting $\{\omega_i^d\}$, implicitly ensuring the robustness of the learned data term.

Table 7. *Parameters of the pixel-dependent weight prediction network for the data term in the SIMAP baseline. CR* denotes a convolutional layer followed by a ReLU. *CS* denotes a convolutional layer followed by a sigmoid function. We denote the number of filters for the data term as M.

Layers	Kernel size	Number of features	Stride
CR_1	3×3	64	1
CR_2	3×3	64	1
CR_3	3×3	64	1
CR_4	3×3	64	1
CR_5	3×3	64	1
CS	3×3	M	1

Table 8. *Model size comparisons of our learned spatially-variant MAP model (SVMAP) and the learned spatially-invariant baseline model (SIMAP).* M and N denote the number of filters for the data and regularization terms, respectively.

	SIMAP	SVMAP (*ours*)
(M, N)	$(3, 5)$	$(3, 5)$
Total parameters (M)	1.92	1.29

Table 9. *Effectiveness of predicting spatially-variant filters.* All methods are evaluated on the dataset of [24] with 1% Gaussian noise (kernel size from 13×13 to 35×35 pixels). M and N denote the number of filters for the data and regularization terms, respectively.

	SIMAP			SVMAP (*ours*)
(M, N)	$(3, 5)$	$(9, 15)$	$(18, 30)$	$(3, 5)$
PSNR (dB) / SSIM	31.25/0.8861	31.36/0.8887	31.40/0.8900	**31.89/0.8973**

Next, we compare the model size of our spatially-variant model and the spatially-invariant baseline method SIMAP in Tab. 8. As the spatially-invariant baseline SIMAP needs to predict both the filters $\{f_i, g_j\}$ and the weights $\{\omega_i^d, \omega_j^r\}$, the total number of parameters is actually larger than ours. In addition, as shown in Tab. 4 of the main paper, our approach performs notably better than the SIMAP baseline, which demonstrates the effectiveness of predicting spatially-variant filters within the MAP framework.

We further evaluate whether predicting more spatially-invariant filters can achieve a similar performance as predicting spatially-variant filters. Table 9 shows that using more spatially-invariant filters (SIMAP with $M = 9$, $N = 15$ or with $M = 18$, $N = 30$) can slightly improve the results, but still performs worse than our spatially-variant formulation (SVMAP with $M = 3$, $N = 5$).

B.2. Iteration-dependent *vs.* indenpendent filters

As stated in Sec. 3.3 of the main paper, we use the same network architecture in different IRLS iterations, but the network parameters are *not* shared across iterations. We further compare with a baseline method (*SVMAP$_{share}$* for short) that shares the network parameters across various IRLS iterations and is implemented in the same way as ours otherwise. Table 10 shows that our approach achieves better image quality than the baseline method with iteration-independent filters, which demonstrates the effectiveness of predicting iteration-dependent filters.

Table 10. *Effectiveness of predicting iteration-dependent filters.* All methods are evaluated on the dataset of [24] with 1% Gaussian noise (kernel size from 13×13 to 35×35 pixels).

	SVMAP$_{\text{share}}$	SVMAP (*ours*)
PSNR (dB) / SSIM	31.64/0.8929	**31.89/0.8973**

Table 11. *Effectiveness of the number of updating iterations in IRLS.* All methods are evaluated on the dataset of [24] with 1% Gaussian noise (kernel size from 13×13 to 35×35 pixels).

Iterations	1	2	3	50
[21]	24.11/0.6308	25.33/0.6946	26.15/0.7301	29.21/0.8260
SVMAP (*ours*)	31.47/0.8889	31.89/0.8973	32.06/0.8993	–

B.3. Effect of the number of updating iterations

Since we use the IRLS method to solve the problem in Eq. (10b) of the main paper, one needs to iteratively update the filters and estimate the latent image. Since we use deep neural networks to predict the filters, we do not need as many iterations as classical IRLS methods. To verify this, we evaluate the effect of the number of iterations by varying it from 1 to 3. In addition, we also compare with the method of Levin *et al*. [21], which uses a classical IRLS approach. Table 11 shows that our method with 2 updating iterations can already achieve good results and using more iterations does not improve the image quality very significantly. In contrast, [21] needs about 50 iterations to reach convergence. Thus, directly estimating the components in the IRLS method can significantly improve the image quality and reduce the number of required iterations.

B.4. Model size and run-time

Next, we compare the model size and the average run-time of our approach to a selection of representative methods in Tab. 12. We benchmark the run-time on a machine with an Intel Xeon E5-2650 v4 CPU and an NVIDIA TITAN Xp GPU. The proposed approach tends to have more parameters than previous work, as the network for predicting the spatially-variant filters does not share its parameters across the various IRLS iterations. However, the average run-time of our approach is comparable or even less than other evaluated methods with smaller model sizes. Note that the methods [12, 49, 50] need several iterations to achieve reasonable accuracy, *e.g*. [12, 50] set the number of iterations to 30. In contrast, by integrating the MAP-based optimization framework as a constraint, we develop an end-to-end network to learn spatially-variant iteration-dependent filters for each IRLS iteration. As demonstrated in Sec. B.3

Table 12. *Model size and run-time.* We evaluate all methods on the same machine in the same settings. The average run-time is tested on images from [41] with $800 \times 1024 \times 3$ pixels.

	FCN [49]	IRCNN [50]	RGDN [12]	SVMAP
Total parameters (M)	0.45	0.19	1.26	1.29
Run-time (s)	2.81	4.59	11.27	2.94

and Tab. 11, our approach does not need as many iterations as classical IRLS methods and using 2 updating iterations can already achieve good results. Thus, the proposed approach is efficient with favorable accuracy.

B.5. Visualization of predicted spatially-variant filters

To intuitively illustrate what spatially-varying filters the network learns to predict, we show the visualization of some predicted filters in Fig. 7. The blurry input and ground truth are shown in Fig. 7(a) and (g). Figure 7(b) is the initial result used to bootstrap the IRLS algorithm; this initialization is obtained by deconvolving with an ℓ_2 norm-based data term and a Gaussian prior. As the ℓ_2 data term is not robust to outliers and the Gaussian prior cannot fully capture the characteristic properties of clear images, the deconvolved result in Fig. 7(b) contains significant artifacts in the saturated areas and the fine-scale structures are not recovered well. Figure 7(e) and (h) are the predicted spatially-variant filters for the data term and the regularization term, which are predicted from Fig. 7(b). For better understanding, we show all the predicted filters in an image, *i.e*. for each pixel in the image, a filter of $s_f \times s_f$ pixels (for the data term) or $s_g \times s_g$ pixels (for the regularization term) is shown.

As shown in Fig. 7(e) and (h), both the filters predicted for the data term and the regularization term can effectively capture the spatially-variant image characteristics. Specifically, since the data term measures the goodness-of-fit, the filters f_i predicted for the data term vary depending on the image reconstruction error (in the sense of Eq. (5) of the main paper). We note that the initial latent image in Fig. 7(b) exhibits significant errors in saturated areas. Comparing this to the predicted filters in Fig. 7(e), we observe that the trained network predicts quite different filters for saturated areas (with larger reconstruction errors) than in non-saturated areas (with smaller reconstruction errors). This improves the quality of the latent image (Fig. 7(c)), but saturated pixels continue to violate the underlying convolutional assumption of the data term (as stated in Sec. 3.1 of the main paper). Thus, even further iterations show different filters being predicted for areas with and without saturation (Fig. 7(f)). Similarly, the predicted filters g_j for the regularization term are based on the latent image and can adapt to the image content. Hence, differing g_j are predicted for different image structures, *e.g*., flat and textured areas in Fig. 7(h) and (i). Finally, our approach generates a better deblurred image with finer structures and detail (Fig. 7(d)). We additionally visualize the predicted spatially-variant filters on another example in Fig. 8, which shows similar properties of the predicted filters as those in Fig. 7. The comparisons in Figs. 7 and 8 further demonstrate the effectiveness of the proposed spatially-variant MAP model for non-blind image deblurring.

B.6. Our learned regularizer *vs.* the regularizer constructed by pre-trained deep CNNs

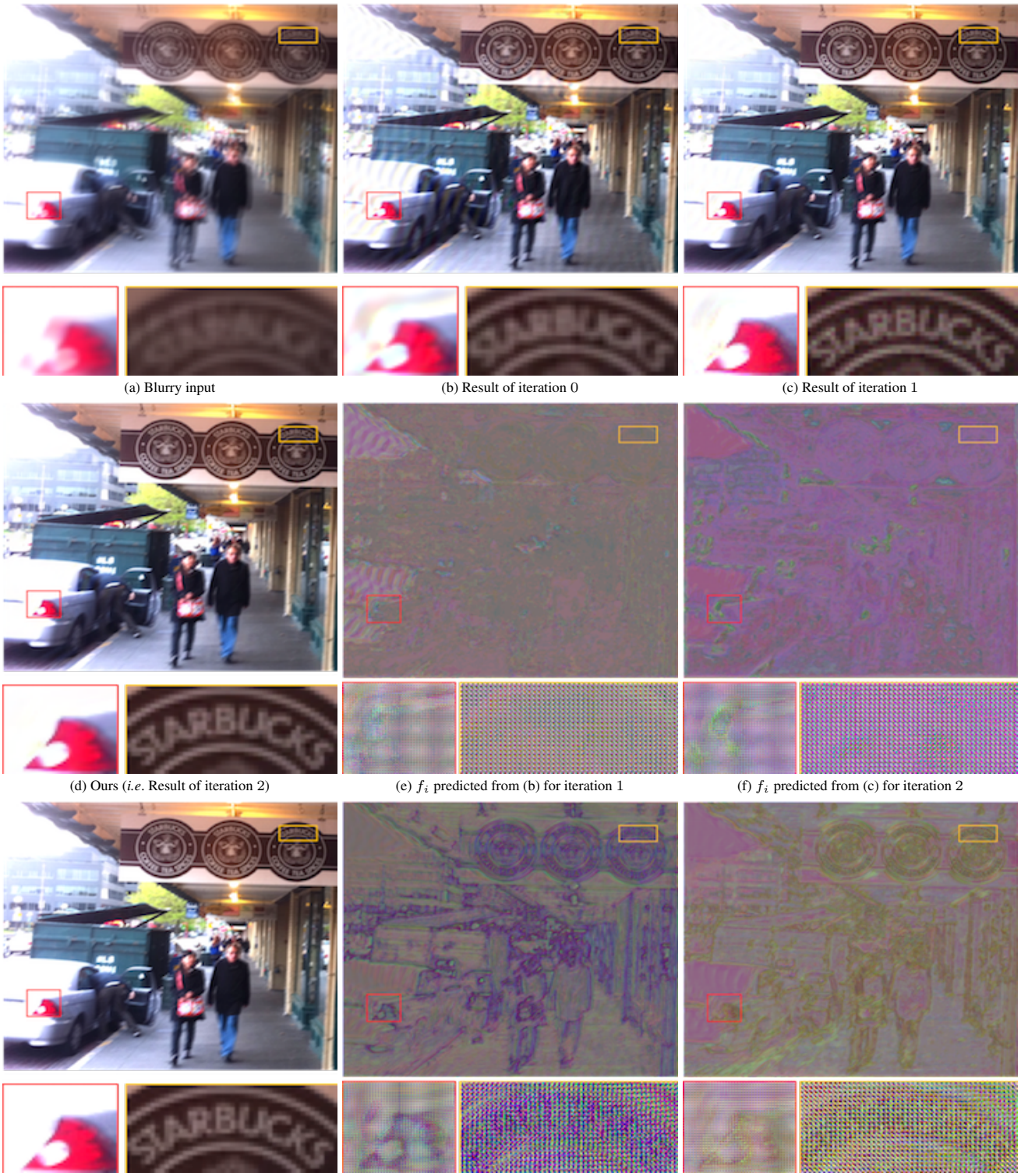
The method of Zhang *et al*. [50] decouples the data term and the regularization term into two individual subproblems and solves the regularization term-related one by a pre-trained CNN denoiser, which contains 7 convolutional layers. In contrast, we jointly learn both the data and regularization terms, where our network involves 6 convolutional layers to predict the filters for constructing the regularizer in Eq. (12). Since our joint learning of both the data and regularization terms takes advantage of their interplay, our approach is able to build more expressive deblurring models. Note that the ℓ_2 data term is theoretically the most suitable one to model the Gaussian noise underlying the dataset used in Tab. 1 of the main paper. However, our approach still performs better than [50] with the ℓ_2 data term, preserving finer detail as shown in Fig. 11 of the main paper, which highlights the effectiveness of jointly learning the data and regularization terms.

B.7. Model generalization ability

Although the proposed method learns an image-adaptive model, we have shown in Sec. 4 of the main paper that our model trained on one image dataset can be applied to images from other datasets. Specifically, to train the model for handling blurry images with Gaussian noise, we use the training datasets from [23, 24]. Then we evaluated our trained model on the test datasets of [24, 41] in Tab. 1, where the dataset by Sun et al. [41] is a dataset different from the training dataset. For handling blurry images with saturated areas, as stated in Sec. 4.1, we use the training data collected from Flickr and test on images from the literature [6, 10, 27, 47], so that the training and test datasets are also different. Thus, our model trained on one image dataset generalizes to other datasets. We further evaluate our approach on several real-world images in Figs. 5, 14 and 15, which are more complex than the images in the training datasets, to demonstrate the generalization ability of our image-adaptive model.

B.8. More experimental results on blurry images with other degradations

In the main paper, we evaluate our model on blurry images with the two most common degradations, *i.e*. Gaussian



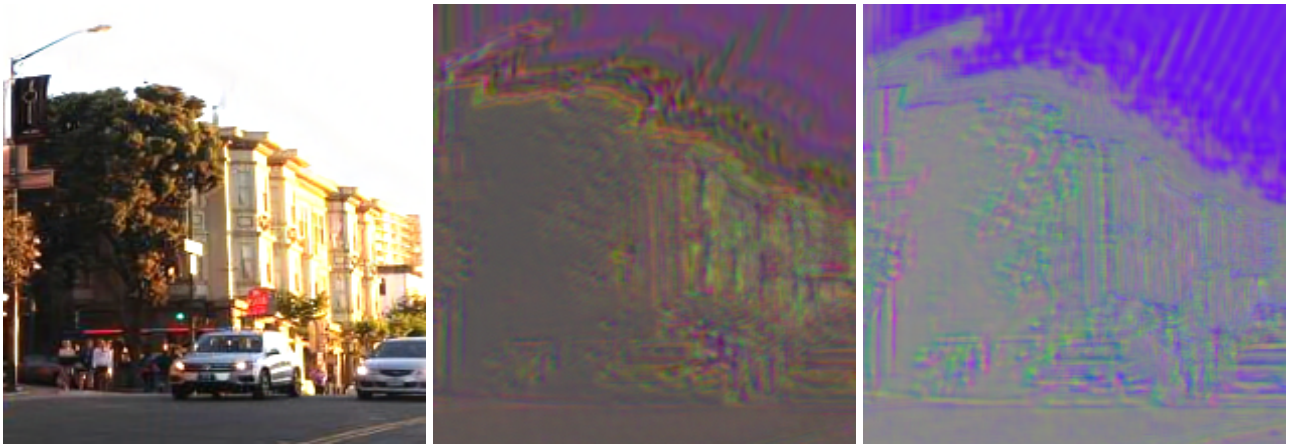
(a) Blurry input (b) Result of iteration 0 (c) Result of iteration 1

(d) Ours (*i.e.* Result of iteration 2) (e) f_i predicted from (b) for iteration 1 (f) f_i predicted from (c) for iteration 2

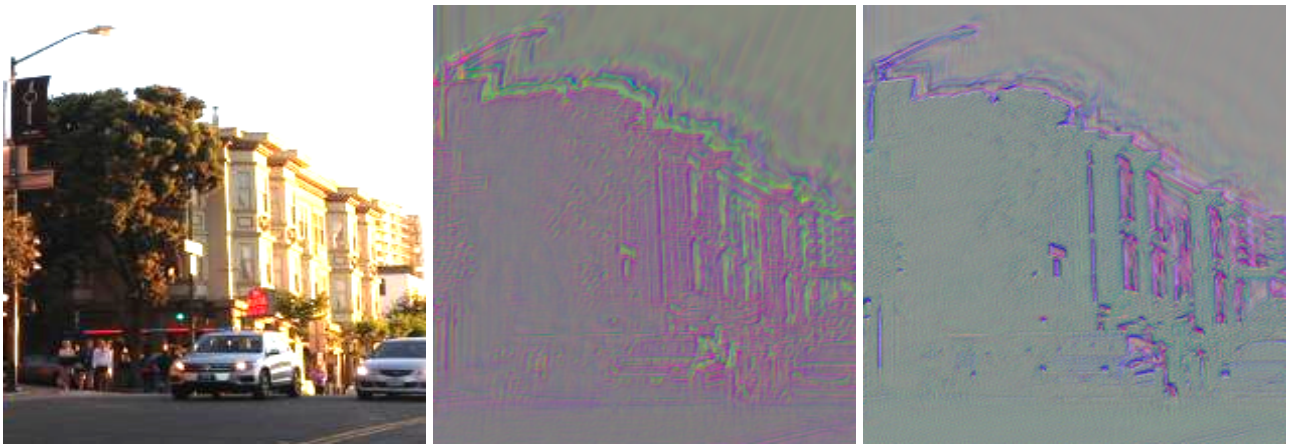
(g) Ground truth (h) g_j predicted from (b) for iteration 1 (i) g_j predicted from (c) for iteration 2

Figure 7. *Visualization of predicted spatially-variant filters.* By predicting spatially-variant filters for both the data term *(e), (f)* and the regularization term *(h), (i)*, our approach can effectively leverage pixel-dependent properties of the image structure and generate a much clearer image with finer detail as shown in *(d)*. (Best viewed on high-resolution display with zoom-in.)

(a) Blurry input (b) Result of iteration 0 (c) Result of iteration 1



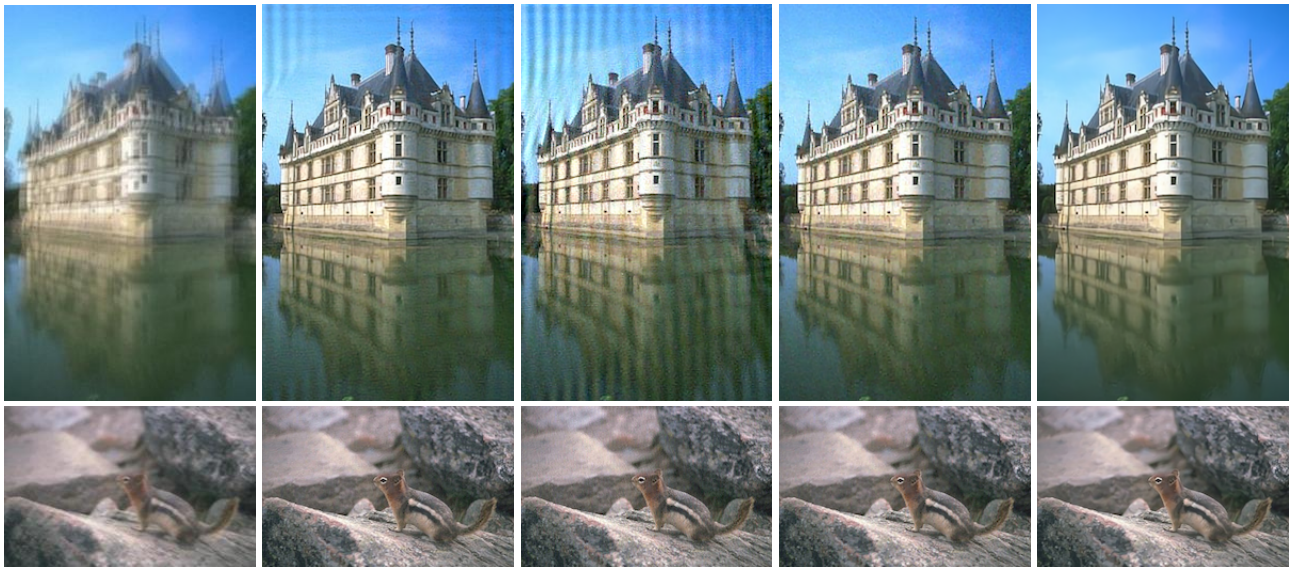
(d) Ours (*i.e.* Result of iteration 2) (e) f_i predicted from (b) for iteration 1 (f) f_i predicted from (c) for iteration 2



(g) Ground truth (h) g_j predicted from (b) for iteration 1 (i) g_j predicted from (c) for iteration 2

Figure 8. *Visualization of predicted spatially-variant filters on another example.* By predicting spatially-variant filters for both the data term *(e), (f)* and the regularization term *(h), (i)*, our approach can effectively leverage pixel-dependent properties of image structure and generate a much clearer image with finer-detail in *(d)*. (Best viewed on high-resolution display with zoom-in.)

noise and saturated pixels. In this section, we provide more experimental results on blurry images with other degradations. Table 13 shows a quantitative comparison on the dataset of [24], where the blurry images have 1% Gaussian noise added and were then JPEG compressed with a quality parameter of 90. Our approach performs substantially better than the competing methods. Figure 9 shows some visual comparisons, where [6, 11, 50] suffer from vis-



(a) Blurry input (b) IRCNN [50] (c) LDT [11] (d) Cho et al. [6] (e) SVMAP *(ours)*

Figure 9. *Example with simulated blur, Gassian noise, and JPEG compression from the dataset of [24].* The methods [6, 11, 50] cannot effectively generate clear images and their results contain severe noise and artifacts. Compared to these methods, our SVMAP approach recovers a much clearer image with finer detail. (Best viewed on high-resolution display with zoom-in.)

(a) Blurry input (b) Xu et al. [54] (c) LDT [11] (d) Cho et al. [6] (e) SVMAP *(ours)*

Figure 10. *Example with simulated blur and impulse noise from the dataset of [41].* The results from [6, 11, 54] generally have less detail and partly exhibit artifacts. In contrast, our SVMAP approach recovers a clear image with fine structures. (Best viewed on high-resolution display with zoom-in.)

Table 13. *Quantitative comparison to state-of-the-art methods* on the dataset of [24] with Gaussian noise and JPEG compression.

	IRCNN [50]	LDT [11]	Cho [6]	SVMAP
PSNR(dB) / SSIM	28.79/0.8209	26.41/0.7389	28.64/0.8205	**30.05/0.8604**

ible noise and artifacts. In contrast, our method is able to recover much clearer images.

We additionally show some visual comparisons in Fig. 10 on the dataset of [41] with 1% impulse noise added. Our approach achieves favorable performance against state-of-the-art methods [6, 11, 54] that are specialized in and effective for handling impulse noise. As shown, our method is effective in preserving fine detail and obtains a clear image despite not being specialized to this task, supporting the observation that our spatially-variant MAP model can adapt to different scenarios.

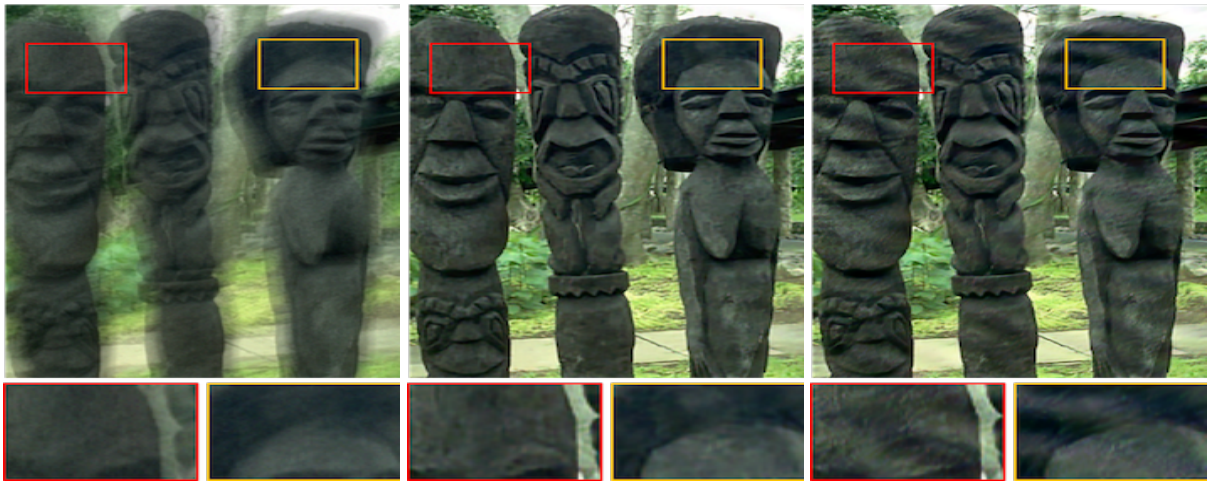
C. Qualitative Comparisons

We finally provide additional visual comparisons with various competitive methods [11, 12, 18, 36, 38, 39, 49, 50, 52] including the previous state of the art on images with both simulated blur (Figs. 11 to 13) and real camera shake (Figs. 14 and 15).

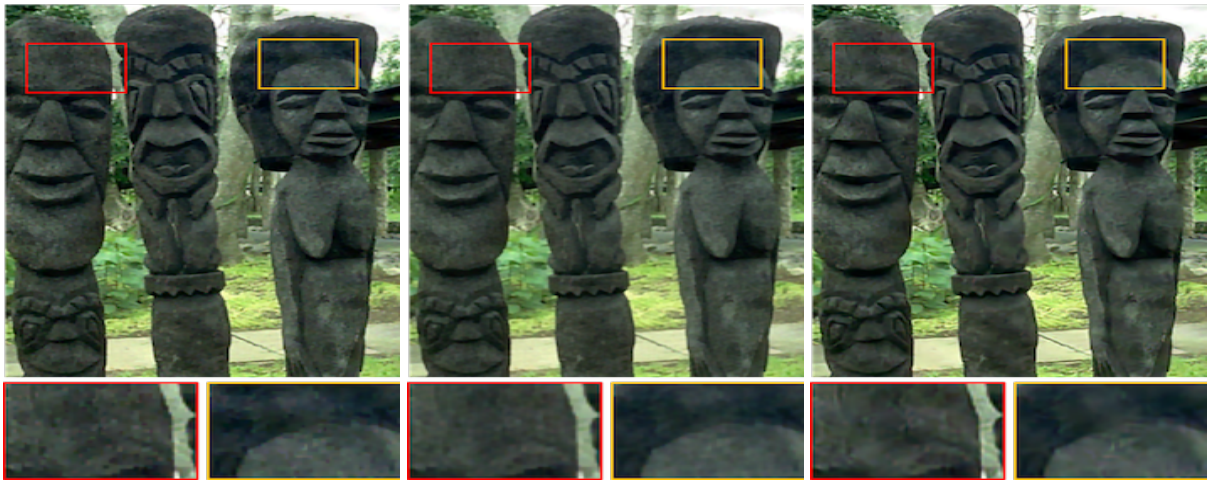
References

[53] Jinshan Pan, Deqing Sun, Hanspeter Pfister, and Ming-Hsuan Yang. Deblurring images via dark channel prior. *IEEE TPAMI*, 40(10):2315–2328, 2017. 10, 11

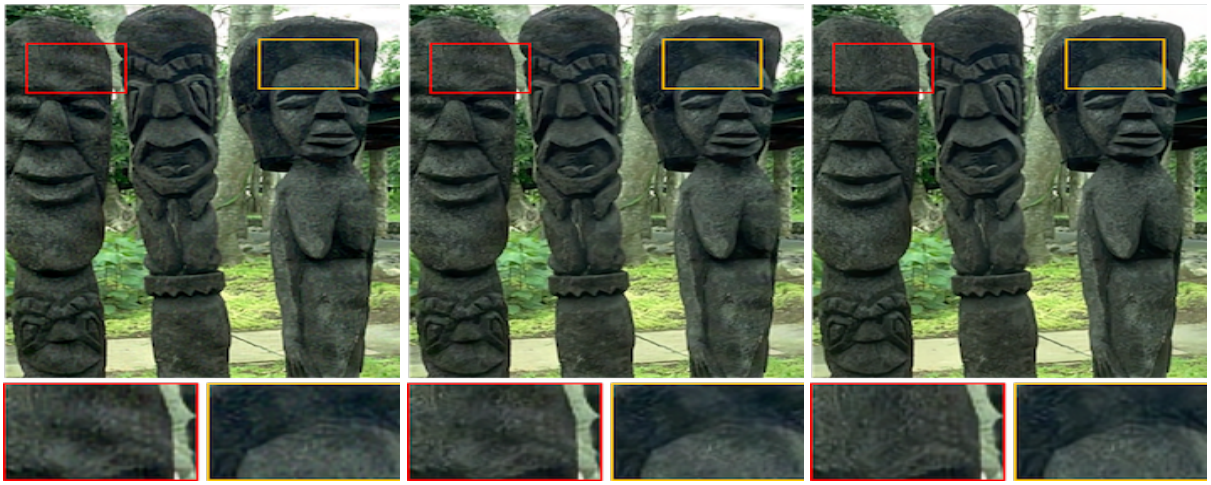
[54] Li Xu and Jiaya Jia. Two-phase kernel estimation for robust motion deblurring. In *ECCV*, volume 1, pages 157–170, 2010. 6, 11



(a) Blurry input (b) EPLL [52] (c) MLP [38]

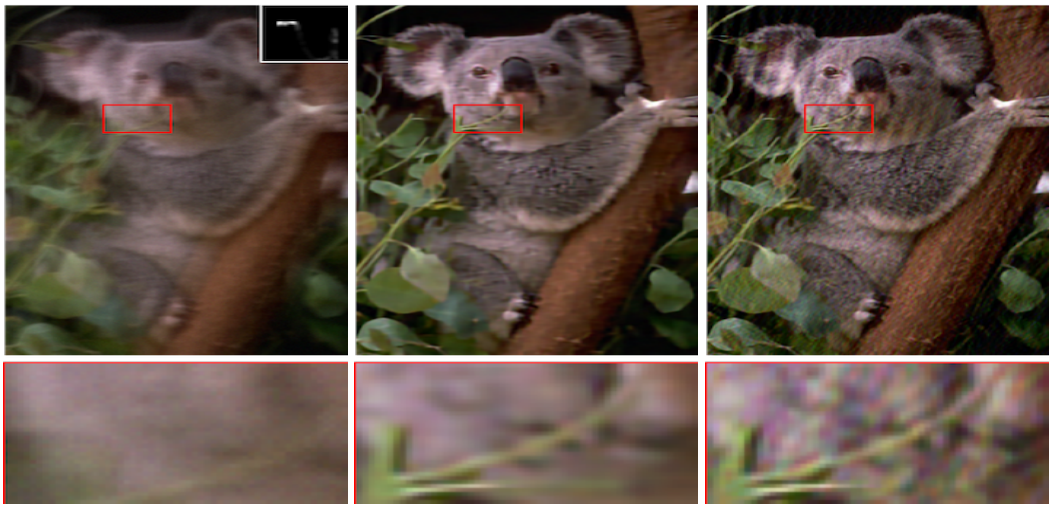


(d) CSF [36] (e) LDT [11] (f) FCN [49]

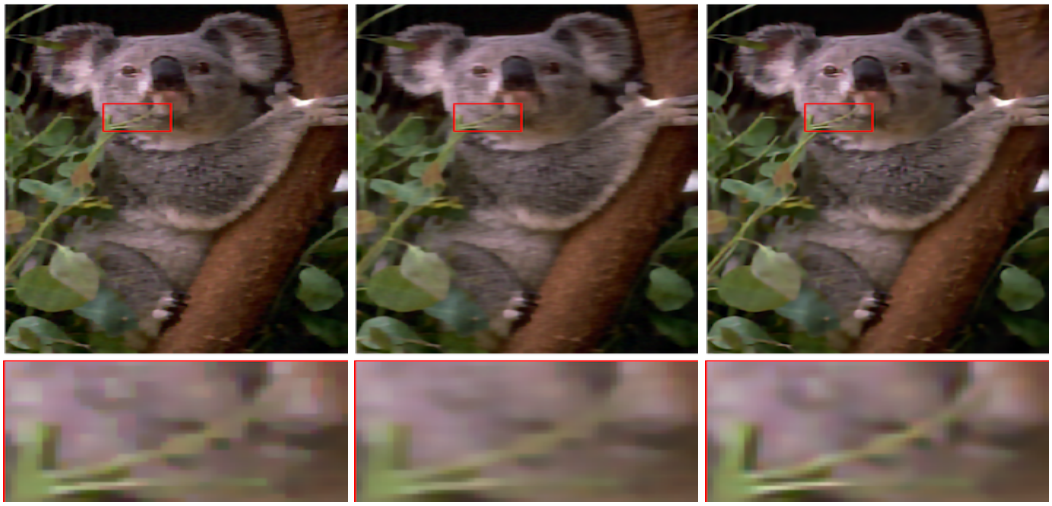


(g) IRCNN [50] (h) FNBD [39] (i) SVMAP *(ours)*

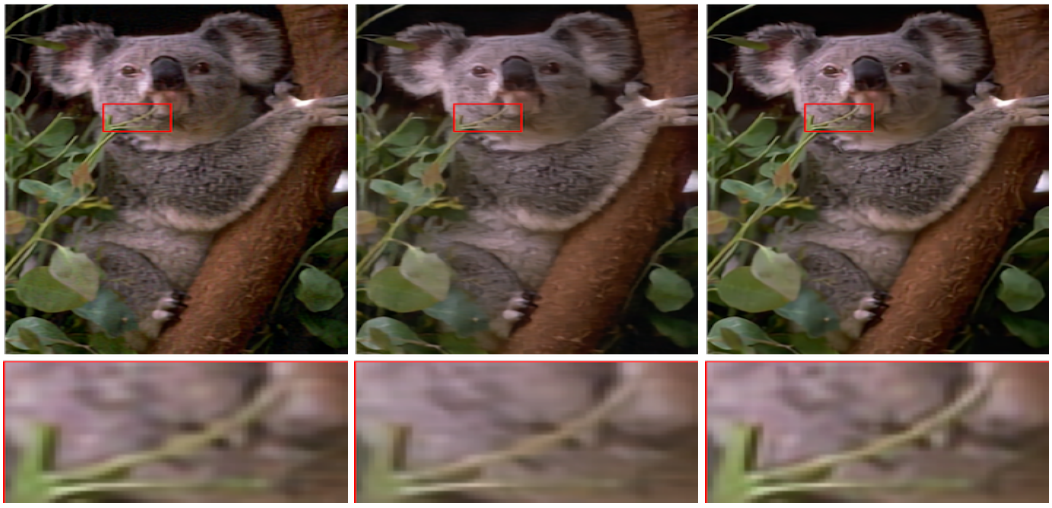
Figure 11. *Example with simulated blur (*1% *noise level) from the dataset of [24].* The results by [38, 39, 50] contain significant artifacts in *(c), (g)*, and *(h)*. The methods [11, 36, 49, 52] oversmooth fine-scale structures in *(b)* and *(d)–(f)*. Compared to existing methods, our SVMAP approach can effectively preserve finer detail as shown in *(i)*. (Best viewed on high-resolution display with zoom-in.)



(a) Blurry input (b) EPLL [52] (c) MLP [38]



(d) CSF [36] (e) LDT [11] (f) FCN [49]



(g) IRCNN [50] (h) RGDN [12] (i) SVMAP *(ours)*

Figure 12. *Example with simulated blur (*1% *noise level) from the dataset of [24].* The deblurred image by [38] exhibits severe artifacts in *(c)*. For other methods, fine-scale structures are not effectively recovered in *(b)* and *(d)–(h)*. In contrast, our SVMAP approach can effectively restore a clear image with finer detail as shown in *(i)*. (Best viewed on high-resolution display with zoom-in.)

(a) Blurry input (b) EPLL [52] (c) MLP [38]

(d) CSF [36] (e) LDT [11] (f) FCN [49]

(g) FDN [18] (h) RGDN [12] (i) SVMAP *(ours)*

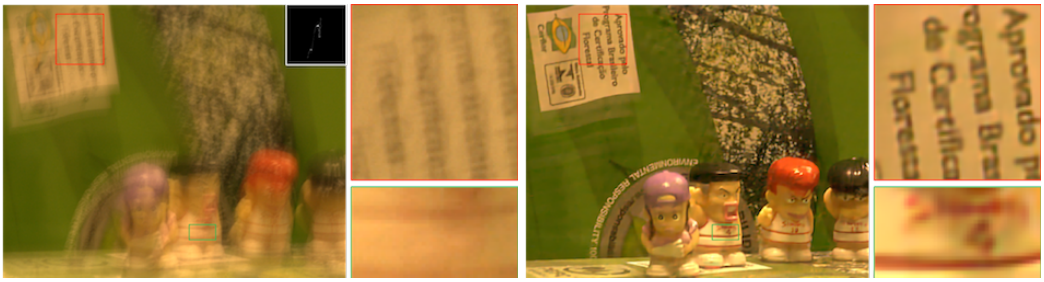
Figure 13. *Example with simulated blur (5% noise level) from the dataset of [41].* The methods [11, 12, 18, 36, 38, 49, 52] do not effectively recover fine-scale structures and detail in *(b)–(h)*. Compared to existing methods, our SVMAP approach can preserve finer detail, *e.g.* the railing and characters in *(i)*.

(a) Blurry input (b) EPLL [52] (c) MLP [38]

(d) CSF [36] (e) LDT [11] (f) FCN [49]

(g) IRCNN [50] (h) FNBD [39] (i) SVMAP *(ours)*

Figure 14. *Example with real camera shake from [53].* The kernel in *(a)* is estimated by the method of [27]. The fine-scale structures in *(b), (e),* and *(f)* are over-smoothed by the methods of [11, 49, 52]. The results generated by [36, 38, 39, 50] have severe artifacts in *(c), (d), (g),* and *(h)*. Compared to the competing methods, our deblurred image in *(i)* is better recovered.

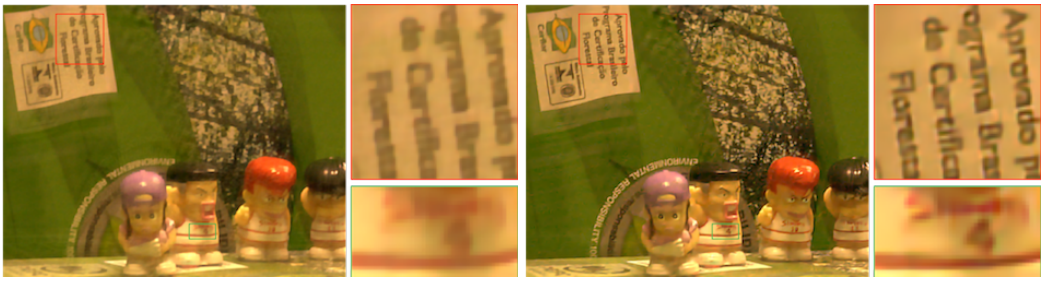


(a) Blurry input

(b) EPLL [52]

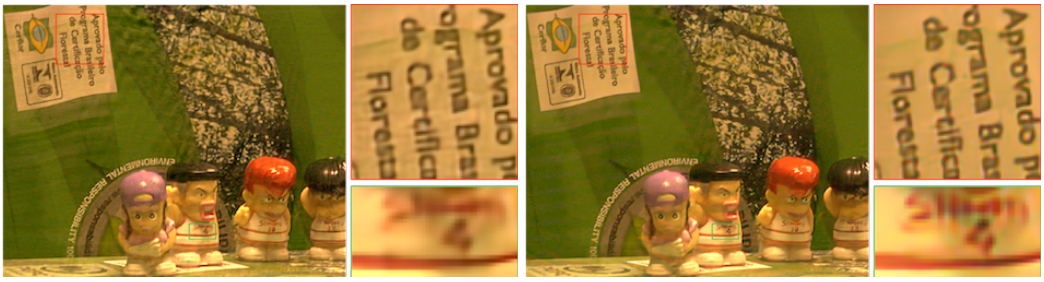
(c) MLP [38]

(d) CSF [36]



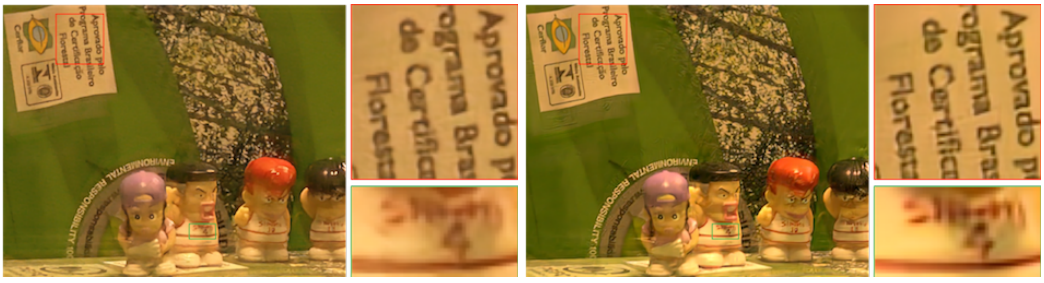
(e) LDT [11]

(f) FCN [49]



(g) IRCNN [50]

(h) FDN [18]



(i) RGDN [12]

(j) SVMAP *(ours)*

Figure 15. *Example with real camera shake from [54].* The kernel in *(a)* is estimated by the method of [53]. The results obtained by the methods [12, 18, 36, 38, 50] contain significant artifacts or color distortions in *(c), (d),* and *(g)–(i).* The methods of [11, 49, 52] do not effectively restore fine-scale structures and detail as shown in *(b), (e),* and *(f).* In contrast, our SVMAP approach generates a much clearer image with finer detail in *(j).*