Deep Model-Based Super-Resolution with Non-uniform Blur

Charles Laroche  
GoPro & MAP5  
charles.laroche@u-paris.fr

Andrés Almansa  
CNRS & Université Paris Cité  
andres.almansa@parisdescartes.fr

Matias Tassano  
Meta Inc.  
mtassano@meta.com

Abstract

We propose a state-of-the-art method for super-resolution with non-uniform blur. Single-image super-resolution methods seek to restore a high-resolution image from blurred, subsampled, and noisy measurements. Despite their impressive performance, existing techniques usually assume a uniform blur kernel. Hence, these techniques do not generalize well to the more general case of non-uniform blur. Instead, in this paper, we address the more realistic and computationally challenging case of spatially-varying blur. To this end, we first propose a fast deep plug-and-play algorithm, based on linearized ADMM splitting techniques, which can solve the super-resolution problem with spatially-varying blur. Second, we unfold our iterative algorithm into a single network and train it end-to-end. In this way, we overcome the intricacy of manually tuning the parameters involved in the optimization scheme. Our algorithm presents remarkable performance and generalizes well after a single training to a large family of spatially-varying blur kernels, noise levels and scale factors.

1. Introduction

Single image super-resolution (SISR) methods aim to up-sample a blurred, noisy and possibly aliased low-resolution (LR) image into a high-resolution (HR) one. In other words, the goal of SISR is to enlarge an image by a given scale factor $s > 1$ in a way that makes fine details more clearly visible. The problem is ill-posed since there exist many ways to up-sample each low-resolution pixel. In order to further constrain the solution, a prior is usually imposed on the reconstructed HR output via a regularizer. Early Bayesian and variational approaches to the SISR problem used Tikhonov [44, 55], TV [1, 39], wavelet-$\ell_1$ [16], non-local [48], or patch-recurrence [42, 43] regularization schemes, or adaptive filtering techniques [49] to impose a reasonable prior on the HR solution. But classical regularization schemes have shown their limits. In order to cope with real-world SISR problems, which are more ill-posed (higher noise levels, larger zoom factors, larger and more complex blur kernels) recent methods have turned to more powerful deep-learning-based regularizers, regressors or (conditional) generative models. And they succeeded remarkably, producing extremely high-quality results for very large ($\times 16$) scale factors, as long as blur is uniform and small [52].

The focus of this paper is on more realistic cases where blur kernels are non-uniform and much larger and complex, due mainly to motion blur and defocus blur [63, 73]. Such degradations are very common in action cameras where the camera shake leads to spatially-varying motion blur or in microscopy where the lens blur cannot be assumed to be uniform. In such setting, doing super-resolution and deblurring in two steps is sub-optimal since it suffers from error accumulation. Also, the two steps approach does not exploit the correlation between the two tasks. Those observations raise the need for a super-resolution model robust to spatially-varying blur. This particular case received much less attention in the recent deep-learning-based SISR literature. Among recent works, BlindSR [8] can handle non-uniform blur, but it does so only for relatively small and isotropic blur kernels which are quite far from real-world examples. Other models such as USRNet [69], can handle larger, anisotropic motion blur kernels, but they fail to generalize to spatially-varying blur. This paper brings together these two characteristics to propose the first deep-learning based SISR method that can deal with both spatially-varying and highly anisotropic, complex blur ker-
nels.

Like in [69], our architecture is an unfolded version of an iterative optimization algorithm that solves the underlying posterior maximization problem. As demonstrated in [69], this kind of model-based architecture provides a remarkable ability to generalize to a large family of blur kernels. In order to allow our architecture to deal with spatially-varying blur, while keeping computational complexity low, we derive the unfolding from a linearized version of the ADMM algorithm.

The rest of the paper is organized as follows: In Section 2 we review the recent SISR literature with an emphasis in their support for spatially-varying and highly-anisotropic blur kernels. Figure 2b summarizes our review which is later refined in Section 4.2. Section 3 introduces our architecture, its relationship to deep Plug & Play, and linearized ADMM algorithms and provides details on how our end-to-end architecture has been trained. The extensive experimental evaluation in Section 4 shows that our model significantly improves state-of-the-art performance on super-resolution with non-stationary blur, and that it can easily generalize to various non-uniform blur kernels, up-scaling factors, and noise levels which is interesting for real-world applications. Training code and pre-trained model are available at: https://github.com/claroche-r/DMBSR

2. Related Work

2.1. Learning Based Super-Resolution

Several deep learning-based methods have been proposed to approach the SISR problem. SRCNN [11] is among the first models of this type. They employed a CNN to learn the mapping between an LR image and its HR version. Other methods used a similar approach but modified the architectures or losses [26, 27, 34, 56, 64]. ESRGAN [62] introduced a GAN based loss to reconstruct high-frequency details along with an architecture based on the Residual in Residual Dense Block (RRDB). ESRGAN generates sharp and highly realistic super-resolution on synthetic data. However, it struggles to generalize on real images, as their training dataset is built using bicubic downsampling. To overcome this limitation, BSRGAN [68] proposes to retrain ESRGAN on a more realistic degradation pipeline. Unlike previous SISR methods which disregard the blur kernel, SRMD [70] proposes to give the blur kernel information as an additional input to the network. This method then belongs to the family of so-called non-blind SISR methods. Blind methods such as [2, 8, 23] tackle the issue of kernel estimation for super-resolution using an internal-GAN, iterative kernel refinement or a dedicated discriminator. In [14, 54], the authors address the issue of generalization using an image-specific super-resolver.

2.2. Deep Plug-and-Play

Deep plug-and-play methods can be traced back to [59] where the image restoration problem is solved using ADMM optimization by decoupling the data and regularization terms. Then, they use a denoiser to solve the regularization sub-problem. This idea has been extended to other optimization schemes such as primal-dual methods [24, 40] or fast iterative shrinkage algorithm [25]. A large diversity of denoisers have been used for the regularization. Among trained using cyclic loss on intrinsic patches of the low-resolution image. Recently, SwinIR [32] proposes a swin transformer [38] based architecture that achieves state-of-the-art results while heavily reducing the number of parameters of the network.
them, BM3D has been used the most [9, 24, 25], but more recently deep CNN based denoisers have become very popular [40, 58]. [67] provides an analysis of the efficiency of the different deep denoisers for different image restoration tasks. Deep plug-and-play methods can be used to solve a large variety of image restoration tasks such as Gaussian denoising [5], image deblurring [60] or super-resolution [4]. Theoretical aspects of deep plug-and-play algorithms have also been studied using bounded denoisers assumptions [7] or more recently using denoisers whose residual operators are Lipschitz-continuous [30, 51]. More recently, [69] built a deep unfolding network called USRNet for super-resolution using deep plug-and-play optimization.

2.3. Spatially-Varying Blur

Removing uniform blur is a well-studied problem. Classical methods design natural image priors such as $\ell_1$ [31], $\ell_2$ [50] or hyper-Laplacian [28]. CNN learning-based approaches usually build coarse-to-fine deep learning architectures such as [53], where CNN blocks simulate iterative optimization, or [57] which deblurs the image using a scale-recurrent network. The task becomes much more complex when the blur varies spatially. Early approaches decompose the spatially-varying blur into a finite basis of spatially-uniform blur kernels and their respective spatially-varying weights [45]. This approach drastically reduces the dimension of the blur operator and makes it computable in a reasonable amount of time using Fourier transform. [41] build an alternative model designed for faster computation and apply it to deconvolution of spatially-varying blur. More recently, [15–17] approximate the spatially-varying blur operator in the wavelet basis by a diagonal or sparse operator. Their decomposition allows very efficient computation of the blur operator and its transpose since the structure of the operator allows GPU parallelization. Other approaches such as [13] use HQS splitting to decouple the prior and data term. The data step is computed using an approximation of the inverse blur and the prior step is solved using CNN priors. Jointly solving the non-uniform deblurring and upsampling (super-resolution) problem is a much more challenging task that has been much less studied [8]. Most spatially-variant deblurring methods require the blur operator to be known. Estimating the non-uniform blur kernel has been tackled for several applications such as defocus [20], lens aberration [19] super-resolution [33] and motion blur [6].

3. Model

3.1. Problem Formulation

The standard model for single-image super-resolution with multiple degradations usually assumes that the low-resolution image is a blurry, noisy and subsampled version of a given high-resolution image,

$$y = (x \circ k) \downarrow_s + \epsilon \text{ with } \epsilon \sim \mathcal{N}(0, \sigma^2),$$  

(1)

with $x$ the high-resolution image, $y$ its low-resolution version, $k$ the blur kernel, $\downarrow_s$ the subsampling operator with scale factor $s$, and $\epsilon$ the noise. This formulation assumes that the blur kernel is uniform all over the image which makes the computation of the low-resolution image fast using convolution or fast Fourier transform. This assumption is not always realistic since camera or object motion will often result in non-uniform blur as illustrated in Figure 2a. In this example, background objects are moving with respect to the camera, so they appear blurry, whereas foreground objects are sharp. Spatially-varying blur can also appear when the objects are out-of-focus. In this case, the blur is closely related to the depth of field. Taking into account spatially-varying blur, the degradation model in Equation (1) replaces the convolution operator with a more general blur operator that varies across the pixels

$$y = (Hx) \downarrow_s + \epsilon \text{ with } \epsilon \sim \mathcal{N}(0, \sigma^2),$$  

(2)

where $Hx$ corresponds to the non-uniform blur operator applied to image $x$ (flattened as a column vector). Working with unconstrained $H$ leads to computationally expensive operations. In our work, the only restriction we make on $H$ is that $Hx$ and $H^T x$ must be computed in a reasonable amount of time. A basic example for such a use case is the O’Leary model [45] where $H$ is decomposed as a linear combination

$$H = \sum_{i=1}^{P} U_i K_i$$  

(3)

of uniform blur (convolution) operators $K_i$ with spatially varying mixing coefficients, i.e. diagonal matrices $U_i$ such that $\sum_{i=1}^{P} U_i = Id, U_i \geq 0$. This model provides a convenient approximation of a spatially-varying blur operator, by reducing both the memory and computing resources required to store and compute this operator. In practice, $K_i$ may represent a basis of different kinds of (defocus or motion) blur, and the $U_i$ can represent object segmentation masks or sets of pixels with similar blur kernels.

3.2. Deep Plug-and-Play

Model-based variational or Bayesian methods usually solve the SISR problem in Equation (2) by imposing a prior with density $p(x) \propto e^{-\lambda \Phi(x)}$ to the HR image $x$ (common choices for $\Phi$ are Total Variation or $\ell_1$ norm of wavelet coefficients). Then the maximization of the posterior density $p(x|y) \propto p(y|x)p(x)$ leads to the following optimization
problem to compute the MAP estimator:

\[ x^* = \arg \min_{x} \frac{1}{2\sigma^2} \| (Hx) \downarrow_s - y \|^2 + \lambda \Phi(x) \]

\[ = \arg \min_{x} g(Hx) + \lambda \Phi(x). \]  

(4)  

(5)

This family of optimization problems is often solved using iterative alternate minimization schemes, like ADMM [3], which leads to iterating the following steps for \( k = 0, \ldots, N \):

\[ x_{k+1} = \text{prox}_{\lambda f}((v_k - u_k)) = \mathcal{P}_{\sqrt{\lambda f}}(v_k - u_k) \]  

(6)

\[ v_{k+1} = \text{prox}_{\lambda g}(H(x_{k+1} + u_k)) \]  

(7)

\[ u_{k+1} = u_k + (x_{k+1} - v_{k+1}). \]  

(8)

where \( \text{prox}_{\lambda f} \) is the proximal operator of \( \lambda f \) defined by \( \text{prox}_{\lambda f}(v) = \arg \min f(x) + (1/2\lambda)\|x - v\|^2 \) and \( g \) is defined in 5. For convex \( \Phi \), this is known to converge to the solution of Equation (4) as \( k \to \infty \). Deep plug-and-play methods use more sophisticated (possibly non-convex) learned regularizers by simply replacing the regularization step in Equation (6), by a CNN denoiser \( \mathcal{P}_\gamma \) which was pretrained to remove zero-mean Gaussian noise with variance \( \beta^2 \). The convergence of the iterative plug-and-play ADMM scheme still holds in this non-convex case for careful choices of the denoiser and the hyperparameters \((\lambda, \mu, \rho)\) [51].

In the case of a uniform blur, the \( v \)-update can be efficiently computed using the fast Fourier transform [69, 72] since the operator \( H \) is diagonal in the Fourier basis. However, this is no longer the case for spatially-varying blur. Even for simpler use cases such as the O’Leary model from Equations (3) solving the subproblem (7) can be very computationally expensive. To avoid this, linearized ADMM [47, sec 4.4.2] (see also [18, 36, 46, 71]) substitutes the splitting variable \( v = x \) by \( z = Hx \) and introduces the linear approximation \( \|Hx - z\|^2 \approx \mu(H^THx - z^TH^H)z + \frac{\mu}{2}\|x - z\|^2 \) in the augmented Lagrangian, in order to avoid the need to invert \( H \). This approximation corresponds to a linearization of the regularization \( \|Hx - z\|^2 \) with an extra regularization \( \|x - z\|^2 \) that enforces \( z_k \) to be close to the linearization point of application. As a consequence, Equations (6) to (8) are rewritten as follows:

\[ x_{k+1} = \text{prox}_{\lambda f}(z_k - \frac{\mu}{\rho}H^THz_k + u_k) \]

\[ = \mathcal{P}_{\sqrt{\lambda f}}(z_k - \frac{\mu}{\rho}H^THz_k + u_k) \]  

(9)

\[ z_{k+1} = \text{prox}_{\lambda g}(H(x_{k+1} + u_k)) \]  

(10)

\[ u_{k+1} = u_k + Hx_{k+1} - z_{k+1}. \]  

(11)

with \( \mu, \rho \) hyper-parameters of the optimization scheme.

Now Equation (10) can be easily computed in closed form since it does not require to invert a matrix involving \( H \) anymore (see Section 2 of the Supplementary Material for more information). We have:

\[ z_{k+1}(i, j) = \left( \frac{(y_{\downarrow s}) + \sigma^2 \mu(Hx_{k+1} + u_k)(i, j)}{\sigma^2 \mu + \delta_{i=0 (\text{mod } s)} \delta_{j=0 (\text{mod } s)}} \right), \]

(12)

where \( \downarrow_s \) corresponds to the zero-padding up-sampling with scale factor \( s \) and \( \delta_{i=0 (\text{mod } s)} \delta_{j=0 (\text{mod } s)} \) is the indicator function that is equal to 1 on the pixels divided by the scale factor and 0 otherwise. The whole deep plug-and-play iterative program is summarized in Algorithm 1. The linearized ADMM algorithm in Equations (9) to (11) was not yet studied in the Plug & Play context, but recent results in [37] and [22] suggest that it actually converges for careful choices of the parameters \( \lambda, \mu, \rho \) (see Section 1 of the Supplementary Material for details). These theoretical results motivated the Unfolded version of the linearized Plug-and-Play ADMM algorithm that we present in the next section, and is the basis of the experimental results in Section 4.

3.3. Deep Unfolding Networks

Deep plug-and-play methods achieve impressive performance on image restoration tasks. However, their efficiency strongly relies on the choice of their hyper-parameters. Finding correct values for the latter can be challenging. These methods also require a sufficient number of steps to properly converge, which is time-consuming. We improve the runtime and simplify the hyper-parameter tuning process by unfolding our algorithm into a deep learning architecture. This architecture is composed of a fixed and small number of iterations of the linearized ADMM algorithm and a MLP that automatically selects the hyper-parameters. The whole network is optimized

---

Algorithm 1 Deep Plug-and-Play Linearized ADMM algorithm

Solves \( x = \arg \min_x \frac{1}{2\sigma^2} \| (Hx) \downarrow_s - y \|^2 + \lambda \Phi(x) \) using denoiser \( \mathcal{P}_\gamma \) as an implicit regularizer.

**Input:** Measurements \( y \), spatially varying kernel \( H \), scale factor \( s \), noise level \( \sigma \), number of iterations \( N \), hyper-parameters \( \lambda, \mu, \rho \)

**Output:** super-resolved, deconvolved and denoised image \( x_N \)

\[ x_0 \leftarrow y \uparrow_s \]

\[ z_0 \leftarrow Hx_0 \]

\[ u_0 \leftarrow 0 \]

for \( k \in [0, N-1] \) do

\[ x_{k+1} = \mathcal{P}_{\sqrt{\lambda f}}(x_k - \frac{\mu}{\rho}H^TH(x_k - z_k + u_k)) \]

\[ z_{k+1}(i, j) = \frac{(y_{\downarrow s}) + \sigma^2 \mu(Hx_{k+1} + u_k)(i, j)}{\sigma^2 \mu + \delta_{i=0 (\text{mod } s)} \delta_{j=0 (\text{mod } s)}} \]

\[ u_{k+1} = u_k + Hx_{k+1} - z_{k+1} \]

end for
using end-to-end training. Following the linearized ADMM formulation, the architecture alternates between a prior-enforcing step \( P \) corresponding to Equation (9), a data-fitting step \( D \) (see Equation (10)) and finally an update block \( U \) (see Equation (11)). These blocks are respectively stacked \( N \) times corresponding to the number of iterations. The optimization process requires the hyper-parameter triplets \((\lambda, \mu, \rho)\) that are predicted by the hyper-parameters block \( H \) at each step. The resulting architecture of our deep-unfolding network is presented in Figure 3. Next, we present each block in detail.

The first block of our network is the prior module \( P \). As explained in Section 3.2, Equation (9) corresponds to a denoising problem, which is approximated by a CNN denoiser. Based on the work in [69], we use a ResUNet [10] architecture with the denoising level as an extra input for \( P \). All the parameters of the ResUNet are learned during end-to-end training. Following the linearized ADMM formulation, the architecture alternates between a prior-enforcing step \( P \), a data-fitting step \( D \), and an update block \( U \) at each step. The resulting architecture of our deep-unfolding network is presented in Figure 3. Next, we present each block in detail.

The data-term module \( D \) computes the proximal operator of \( \frac{1}{2\sigma^2} ||z - \gamma_k H^T (H x_k - z_k + u_k) ||_2^2 \) at \( z = H x_{k+1} + u_k \). Following Equation (12), we re-write our data-term as:

\[
D(x_{k+1}, H, u_k, y, \sigma, \alpha_k)
\]

3.4. Training

The architecture is trained end-to-end using the L1 loss for 200 epochs. We start with a learning rate of 1e-4 and decrease it every 50 epochs by a factor 0.1. The ResUNet parameters were initialized by a pre-trained model doing so improves the stability of the model during training. We use \( N=8 \) iterations in our unfolded architecture for the experiments. The network is trained using scale factors \( s \in \{1, 2, 3, 4\} \), noise levels \( \sigma \in [0, 25] \) and spatially varying blur kernels composed various motion blurs and Gaussian blurs.
4. Experiments

4.1. Data Generation

Gathering real-world data with spatially-varying blur and their respective kernels is very complicated. Instead, we train our model using synthetic data. For this experiment, we adopted the O’Leary blur model from Equation (3). This blur decomposition covers a large variety of spatially-varying blurs ranging from motion blur to defocus blur. Figure 4 represents an example of synthetic blur obtained using this formulation. For the training and testing, we used COCO dataset [35]. COCO dataset is a well-known large-scale dataset for object detection, segmentation, and image captioning. It is composed of more than 200K images segmented for 80 object categories and 91 stuff categories representing 1.5 million object instances. We use the segmentation masks to blur the objects and the background. We use both Gaussian kernels and motion blur kernels. We build a database of motion blur kernels using [21]. To ensure a smooth and realistic transition between the blurred areas, we blur the borders of the masks so that a mix between blurs occurs at the edges of the objects. We finally normalize the masks so that their sum is equal to 1 for each pixel. After blurring the image, we apply nearest neighbor downsampling with scale factor varying in \{1, 2, 3, 4\} and Gaussian blur with \( \sigma \in [0, 25] \).

Finally, our data generation pipeline implements the following degradation model:

\[
y = \left( \sum_{i=1}^{P} U_{i}K_{i}x \right) \downarrow_{s} + \epsilon, \tag{15}
\]

with \( x \) the clean image, \( y \) its low-resolution version, \( s \) the scale factor, and \( \epsilon \) the noise.

4.2. Compared Methods

We compare the proposed model to Bicubic upsampling (widely used baseline), RRDB [62], SwinIR [32], SFTMD [23], BlindSR [8] and USRNet [69]. Few super-resolution models can generalize to non-uniform blur (see Table 2b). We believe that the models listed above represent the current most pertinent solutions for such a setting. However, using the pre-trained weights from the source code of each model leads to poor performance on our testing dataset since they are trained using uniform blur kernels. In order to ensure a fair comparison with our model, we retrain all those architectures on our database. We use the MSE loss for the retraining of all models so that the PSNR is maximized. Next, we present in details how we use those architectures.

RRDB and SwinIR described in section 2.1 are blind methods so we just retrain them on our dataset using the configuration given by the authors.

SFTMD is the non-blind architecture introduced in [23] that combines kernel encoding using Principal Components Analysis (PCA) and spatial features transforms (SFT) [61] layers. The PCA-encoded blur kernel is fed to the SFT-based network along with the low-resolution image. In the case of spatially-varying blur, they encode the kernels at each pixel’s location and give the resulting spatially-varying map of encoded kernels to the network.

BlindSR proposes an alternative to the PCA for the encoding of the kernel. They use an MLP that is trained along with the super-resolution network. This allows the network to encode more complex kernels. We use the non-blind part of BlindSR that is composed of a backbone with convolutions, dense layers and residual connections with MLP encoding for the kernel.

We finally compare our architecture to USRNet, which is similar to our model but works only with uniform blur. Since we work with the O’Leary model (3), we can apply USRNet on each blurred mask \( U_{i} \) with their corresponding uniform blur kernel \( K_{i} \) and then reconstruct the results by summing the output of the model on each mask. Since we work with the classical USRNet, we do not retrain it and use the weights from the source code of the method. i.e.

\[
USRNet(y, H) \approx \sum_{i=1}^{P} U_{i} USRNet(y, K_{i})
\]

4.3. Quantitative Results

Table 1 summarizes the PSNR, SSIM (structural similarity index) and LPIPS (learned perceptual image patch similarity) on the different testsets. The testsets are constructed from the COCO validation set and the degradation model of Equation 15. Performance on Gaussian and motion blur are evaluated separately. We test the models on x2 and x4 super-resolution without additive noise. Firstly, non-blind models outperform blind ones by a large margin. The extra information about the degradation is well used
Table 1: Quantitative comparison on synthetic data. The displayed metrics correspond respectively to PSNR↑, SSIM↑ and LPIPS↓. Best scores are displayed in red, second best in blue.

<table>
<thead>
<tr>
<th>Scale</th>
<th>Type</th>
<th>Model</th>
<th>Gaussian blur</th>
<th>Motion blur</th>
</tr>
</thead>
<tbody>
<tr>
<td>x2</td>
<td>Blind</td>
<td>Bicubic</td>
<td>(22.52, 0.60, 0.57)</td>
<td>(21.74, 0.62, 0.39)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RRDB [62]</td>
<td>(23.38, 0.67, 0.41)</td>
<td>(23.11, 0.65, 0.36)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SwinIR [32]</td>
<td>(23.47, 0.67, 0.38)</td>
<td>(23.40, 0.67, 0.34)</td>
</tr>
<tr>
<td></td>
<td>Non-blind</td>
<td>SFTMD [23]</td>
<td>(24.76, 0.69, 0.33)</td>
<td>(25.15, 0.74, 0.25)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BlindSR [8]</td>
<td>(26.55, 0.79, 0.24)</td>
<td>(26.40, 0.79, 0.20)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USRNet [69]</td>
<td>(22.64, 0.74, 0.28)</td>
<td>(24.37, 0.75, 0.17)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ours</td>
<td>(36.59, 0.78, 0.26)</td>
<td>(28.20, 0.85, 0.11)</td>
</tr>
<tr>
<td>x4</td>
<td>Blind</td>
<td>Bicubic</td>
<td>(21.81, 0.55, 0.60)</td>
<td>(20.48, 0.56, 0.57)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RRDB [62]</td>
<td>(21.82, 0.57, 0.58)</td>
<td>(22.34, 0.60, 0.56)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SwinIR [32]</td>
<td>(23.01, 0.63, 0.44)</td>
<td>(22.70, 0.64, 0.44)</td>
</tr>
<tr>
<td></td>
<td>Non-blind</td>
<td>SFTMD [23]</td>
<td>(23.12, 0.64, 0.41)</td>
<td>(23.97, 0.67, 0.38)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BlindSR [8]</td>
<td>(25.11, 0.72, 0.34)</td>
<td>(24.54, 0.69, 0.35)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USRNet [69]</td>
<td>(24.08, 0.72, 0.32)</td>
<td>(24.67, 0.72, 0.29)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ours</td>
<td>(25.37, 0.73, 0.31)</td>
<td>(25.96, 0.73, 0.28)</td>
</tr>
</tbody>
</table>

Figure 5: Super-resolution with scale factor s=2 on real-world defocused images

Figure 6: Deblurring results on Lai [29] dataset

by the networks. The blind transformers-based architecture of SwinIR is more efficient than the classical RRDB. For the non-blind architectures, we can see the importance of how the blur operator information is given to the network. Specifically, the neural network encoding of the blur operator from BlindSR outmatches by far the PCA encoder from the IKC version of SFTMD. It highlights the fact that the PCA model is not complex enough to capture interesting features of the blur kernels. The BlindSR model performs well on the Gaussian testset but fails to generalize on motion blur. One reason for this is the fact that motion blurs are too complex to be encoded by PCA or a small MLP. The USRNet model reaches good SSIM and LPIPS whilst having low PSNR. This is mostly due to the fact that this model introduces artefacts which are not captured by SSIM or LPIPS. The poor performances of USRNet underline the fact that networks trained on uniform blur cannot naively generalize well to spatially-varying blur even on simple use cases. Finally, our model outperforms all the other methods by an average of 0.15dB for the Gaussian blur testsets and 1.2dB on the motion blur testset. Additionally, our algorithm out-perform all other methods on SSIM and LPIPS, except for the x2 Gaussian blur case. The success of our method first relies on the fact that the kernel information does not need to be encoded to be fed to the model which allows good de-blurring quality of very complex kernels. Also, the deblurring and super-resolution modules are fixed in our architecture which accounts for increased robustness to different blur types. It is worth pointing out that only a single version of our model was used for all the scale factors without the need to retraining.

4.4. Visual Results

Figure 7 shows a visual comparisons of the different models on x2 super-resolution. We excluded RRDB and SFTMD since their performance are outmatched by SwinIR and BlindSR, respectively. We observe that SwinIR produces results that are still blurry. USRNet yields sharp results on the areas where the blur kernels are not mixed (i.e. when the area of a single degradation map is equal to one and all the other are equal to zeros), but introduces artefacts on the edges of the objects since the deblurring task is not linear. BlindSR super-resolves well the images. However, some areas remain blurry especially when there is motion blur. Finally, our model successfully produces a sharp super-resolved image without artefacts. We observe more texture details and sharper edges.

4.5. Real-world images

Testing our method on real-world images requires that we can access to the blur operator associated to the image. The performance of the super-resolution model strongly relies on the accuracy of the blur kernel. Figure 5 shows SR results of the models on real-world defocus blur where the blur operator was estimated using camera properties [12] while Figure 6 displays deblurring results (scale factor s = 1, no SR) where the blur operator was estimated using [6]. In the first example, it can be observed that BlindSR tends to add over-sharpening artefacts in the image while performing super-resolution. SwinIR returns a cleaner image but that is still blurry. On the other hand, our model returns the sharpest result. These results also highlight the good generalization properties of our algorithm as the model was not trained on defocus blur kernels or smooth variations from a kernel to another at the image scale (the borders of the masks used in training are hard). In deblurring, we compared our model to DMPHN [66], MPRNet [65] and the de-
5. Conclusion & Future Research

In this paper, we approach the problem of single-image super-resolution with spatially-varying blur. We propose a deep unfolding architecture that handles various blur kernels, scale factors, and noise levels. Our unfolding architecture derives from a deep plug-and-play algorithm based on the linearized ADMM splitting technique. Our architecture inherits both from the flexibility of plug-and-play algorithms and from the speed and efficiency of learning-based methods through end-to-end training. Experimental results using the O’Leary blur model highlight the superiority of the proposed method in terms of performance and generalization. We also show that the model generalizes well to real-world data using existing kernel estimation methods.

References


