Appendices for: *Unsupervised Imaging Inverse Problems with Diffusion Distribution Matching*

A. Proof of Proposition 1.

We restate the statement of the proposition for completeness.

Proposition A.1: *For any set of forward model parameters ω, let $p_\omega(y) = \int p_\omega(y \mid x)p(x)dx$ where $p_\omega(y \mid x) = \mathcal{N}(y \mid A_\omega x, \sigma^2 I)$. Let ω_* be a specific set of parameters which we consider to be the optimal set. Then, assuming the data covariance $\Sigma = \mathbb{E}_x[xx^\top]$ is invertible, there exists an orthogonal matrix P such that*

$$p_\omega(y) = p_{\omega_*}(y) \implies \mathcal{A}_\omega = \Sigma^{-1/2} P \Sigma^{1/2} \mathcal{A}_{\omega_*} \tag{7}$$

That is, if the probability distributions p_ω and p_{ω_*} are equal, it is possible to identify ω_* up to rotations P.

Proof. Let f be a quadratic function of the form $f(x) = x^\top C x$. Consider the form of the conditional, and perform a change of variables from y to $Ax + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ independent of x to write:

$$\begin{aligned}
0 &= \mathbb{E}_{y\sim p_\omega}[f(y)] - \mathbb{E}_{y\sim p_{\omega_*}}[f(y)] \\
&= \int f(y) p_\omega(y \mid x) p(x) dx dy - \int f(y) p_{\omega_*}(y \mid x) p(x) dx dy \\
&= \int f(\mathcal{A}_\omega x + \epsilon) p(\epsilon) p(x) dx d\epsilon - \int f(\mathcal{A}_{\omega_*} x + \epsilon) p(\epsilon) p(x) dx d\epsilon \\
&= \mathbb{E}_{x\sim p(x),\epsilon}[f(\mathcal{A}_\omega x + \epsilon) - f(\mathcal{A}_{\omega_*} x + \epsilon)].
\end{aligned}$$

Now

$$\begin{aligned}
\mathbb{E}_{x\sim p(x),\epsilon}[f(\mathcal{A}_\omega x + \epsilon) - f(\mathcal{A}_{\omega_*} x + \epsilon)] &= \mathbb{E}_{x\sim p(x),\epsilon}[(\mathcal{A}_\omega x + \epsilon)^\top C (\mathcal{A}_\omega x + \epsilon) - (\mathcal{A}_{\omega_*} x + \epsilon)^\top C (\mathcal{A}_{\omega_*} x + \epsilon)] \\
&= \mathbb{E}_{x,\epsilon}[x^\top \mathcal{A}_\omega^\top C \mathcal{A}_\omega x - x^\top \mathcal{A}_{\omega_*}^\top C \mathcal{A}_{\omega_*} x + 2x^\top \mathcal{A}_\omega^\top C \epsilon - 2x^\top \mathcal{A}_{\omega_*}^\top C \epsilon] \\
&= \mathbb{E}_x[x^\top \mathcal{A}_\omega^\top C \mathcal{A}_\omega x - x^\top \mathcal{A}_{\omega_*}^\top C \mathcal{A}_{\omega_*} x] \\
&= \mathbb{E}_x[\mathrm{tr}(x^\top \mathcal{A}_\omega^\top C \mathcal{A}_\omega x) - \mathrm{tr}(x^\top \mathcal{A}_{\omega_*}^\top C \mathcal{A}_{\omega_*} x)] \\
&= \mathbb{E}_x[\mathrm{tr}(\mathcal{A}_\omega x x^\top \mathcal{A}_\omega^\top C) - \mathrm{tr}(\mathcal{A}_{\omega_*} x x^\top \mathcal{A}_{\omega_*}^\top C)] \\
&= \langle (\mathcal{A}_\omega \Sigma \mathcal{A}_\omega^\top - \mathcal{A}_{\omega_*} \Sigma \mathcal{A}_{\omega_*}^\top), C \rangle_F = 0 \\
&\implies \mathcal{A}_\omega \Sigma \mathcal{A}_\omega^\top - \mathcal{A}_{\omega_*} \Sigma \mathcal{A}_{\omega_*}^\top = 0
\end{aligned} \tag{8}$$

where $\Sigma = \mathbb{E}_x[xx^\top]$ is the data covariance. Eq. (8) holds since C can be any matrix. Now since Σ is invertible, we can take $\tilde{\mathcal{A}_\omega} = \mathcal{A}_\omega \Sigma^{1/2}$ and $\tilde{\mathcal{A}_{\omega_*}} = \mathcal{A}_{\omega_*} \Sigma^{1/2}$ and we have that $\tilde{\mathcal{A}_\omega} \tilde{\mathcal{A}_\omega}^\top = \tilde{\mathcal{A}_{\omega_*}} \tilde{\mathcal{A}_{\omega_*}}^\top$. Now looking at the SVD of $\tilde{\mathcal{A}_\omega} = U_\omega S_\omega V_\omega^\top$ and $\tilde{\mathcal{A}_{\omega_*}} = U_{\omega_*} S_{\omega_*} V_{\omega_*}^\top$ it holds that

$$U_\omega S_\omega^2 U_\omega^\top = U_{\omega_*} S_{\omega_*}^2 U_{\omega_*}^\top \tag{9}$$

which implies that both $U_\omega = U_{\omega_*}$ and $S_\omega = S_{\omega_*}$. Take P the orthonormal change of basis from V_ω to V_{ω_*}, then

$$\tilde{\mathcal{A}_\omega} = \sum_i \sigma_\omega^{(i)} u_\omega^{(i)} v_\omega^{(i)^*} P = \sum_i \sigma_{\omega_*}^{(i)} u_{\omega_*}^{(i)} v_\omega^{(i)^*} P = \sum_i \sigma_{\omega_*}^{(i)} u_{\omega_*}^{(i)} v_{\omega_*}^{(i)^*} = \tilde{\mathcal{A}_{\omega_*}} \tag{10}$$

and going back to the original operators, $\mathcal{A}_\omega = \mathcal{A}_{\omega_*} \Sigma^{1/2} P \Sigma^{-1/2}$ with P an orthogonal matrix. □

B. Experiment Details

For all experiments we started from the diffusion model implementation EDM2 [20], and adapted it to train flow-matching models. The EMA for model weights based on a power-function instead of an exponential and is parameterized in an unconventional way through the relative standard deviation (see Karras et al. [20] for a precised description).

(a) Measurement (b) FastEM-ΠGDM [23] (c) KernelDiff [44] (d) BlindDPS [12] (e) Ours (f) Ground truth

Figure 7. Sample reconstructions on FFHQ (motion-blur kernel). All methods single-image apart from ours.

(a) Measurement (b) BlindDPS [12] (c) Ours (d) Ground truth

Figure 8. Sample reconstructions on FFHQ (gaussian-blur kernel). All methods single-image apart from ours.

B.1. FFHQ experiment

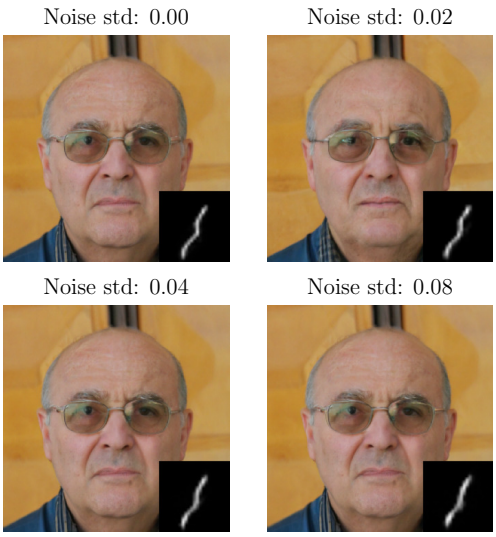
We used a very small CFM model with 4M parameters to train on the first 1000 samples from the FFHQ dataset, downscaled to 256x256, degraded using a fixed motion-blur kernel generated with the code from Borodenko [7] with intensity 0.5 and Gaussian noise with standard deviation 0.02. We trained the model from scratch for 4000 epochs, at the end of which it could generate samples like those in Fig. 2. We then proceeded to the second step of our algorithm by learning a 28x28 blur kernel from 100 different clean samples from FFHQ. We enforced the kernel to sum to 1, and used a sparsity promoting regularizer (simply average of the absolute values of the kernel) with weight 1. We trained the second step for 5000 epochs with a low learning rate (10^{-5} for the blur kernel and 4×10^{-5} for the auxiliary CFM model). Finally we used DiffPIR [62] in the implementation of DeepInv (`https://github.com/deepinv/deepinv`) for reconstructing the full FFHQ test set (1000 images). The baselines were all run from their respective repositories. The only difficulty encountered was with KernelDiff [44] which cannot handle noisy measurements. We thus ran KernelDiff without noise. Figures 7 and 8 compare the image and kernel reconstructions of the different methods for both motion-blur and isotropic Gaussian kernels. Note that the same regularization and other hyperparameters were used to train our method in both experiments.

B.1.1. FFHQ with varying noise levels

The experiments described above and in the main text all used the same random Gaussian noise with standard deviation of 0.02. To determine whether our method is robust to different levels of noise we ran the same experiment with noise-levels 0, 0.04 and 0.08. To compare the models fairly, we then ran DiffPIR using the learned kernels (learned with different noise levels) on data which has been corrupted with the original noise level of 0.02. The training hyper-parameters for all three stages were kept fixed from the previous section. The results of this experiment show a small decrease in accuracy as the noise-level increases, in Sec. B.1.1 support a small deterioration in both kernel estimates and consequently in the reconstruction quality, while remaining fairly robust. Note that the noise is applied to 0-1 normalized images such that a noise level of 0.08 corresponds to 8%, which is very clearly noticeable in the corrupted images.

B.2. Space-varying blur on DPDD

Here we used a larger CFM model with 52M parameters, pretrained on the DIV2K clean training set. This then helps speed up the first step of our algorithm which consists in fine-tuning the model on the same dataset, degraded using the per-pixel blur kernels, as linearly interpolated from the 8x8 PSFs shown in Fig. 10 and Gaussian noise with standard deviation 0.01. The model was trained on patches of size 128x128 from the degraded DIV2K dataset with learning rate 10^{-4} and exponential moving average over the weights. For the second step, we once again directly optimized the parameters of an 8x8 grid of kernels (the same as the ground-truth). A learning rate scheduler is used for the auxiliary flow matching model (square root scheduling with warmup). A batch size of 20 was used, and we found the batch size to significantly affect convergence speed (i.e. convergence depended on how many times each of the two models was updated, so higher batch sizes did not speed up much). For this step, we use a regularization composed of three different terms, the first one is a center term which avoid the kernels to have centering offset, the second one is a sparsity one which is basically a l1



σ_{train}	PSNR	LPIPS	kernel NCC
0	28.93	0.068	0.98
0.02	28.76	0.069	0.94
0.04	28.84	0.070	0.96
0.08	28.01	0.079	0.91

Figure 9. Qualitative and quantitative results for motion-deblurring with increasing amounts of noise. The figure on the left shows the learned kernel and a sample reconstruction with the different noise levels. On the right, the table provides reconstruction metrics and kernel-accuracy metrics for the FFHQ test-set (1000 random images). Note that kernel NCC is the normalized cross-correlation of the learned kernel with the ground-truth kernel.

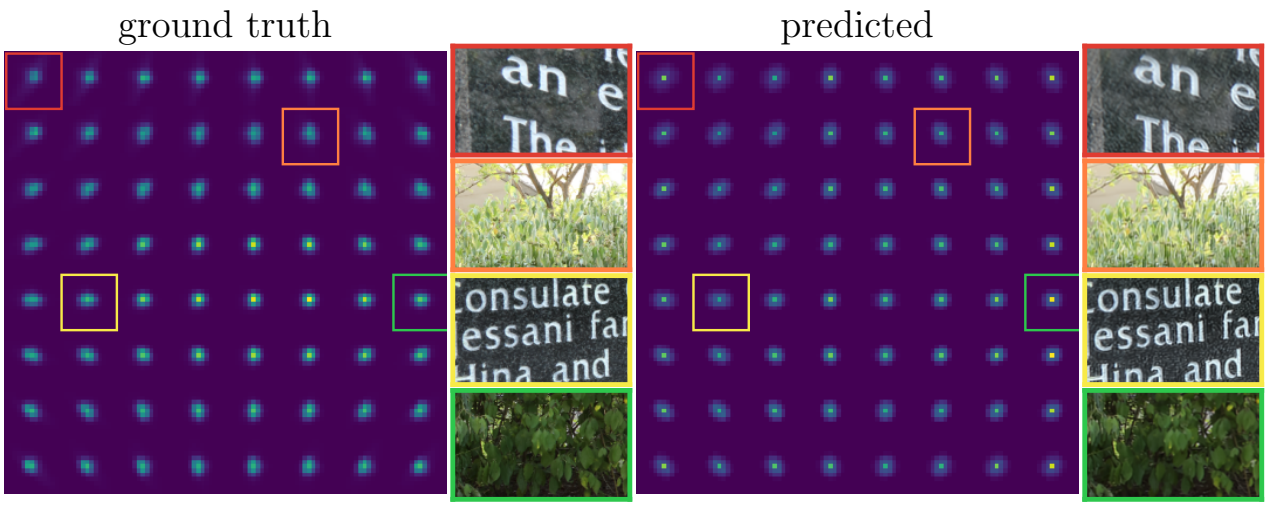


Figure 10. Comparing ground-truth (left) and predicted blur kernels (right), as well as the respective reconstructed images.

regularization which limit the number of non zero pixels in the kernels and the last one is a Gaussian prior which help with the overall shape of kernels. Each term has the same weight of 0.003. For the third step we used the DPDD dataset and perform the forward model inversion in two ways:

- ESRGAN, used by generating a paired dataset on the "train_c" split of DPDD. We used the implementation at https://github.com/XPixelGroup/BasicSR, changed the upscaling factor to 1 since we were not interested in super-resolution and reduced the number of iterations to 60000 since convergence was fast.
- DPIR, again using the implementation from DeepInv, we only need to process the DPDD test-set.

To train the Deflow [51] we use the official repository which is available on github at https://github.com/volflow/DeFlow. Deflow requires a bit of preprocessing to create the training datasets. Indeed, we use both DIV2K and DPDD to train this model. So we use DPDD as a clean dataset with the low quality dataset being the DPDD training set downsampled by 4 using BICUBIC interpolation. And we use DIV2K for the noisy dataset applying a per pixel kernel interpolated from the 8x8 grid on which we add a gaussian noise of standard deviation 0.01. We also downsample the noisy DIV2K to have noisy low quality dataset. Concerning the parameters of the training we only change the batch size in order to make it fit our GPUs and we change the scale to 4 because we only found a pretrained model for this downsampling scale.

(a) DPIR true kernels (b) ESRGAN true kernels (c) INIKNet

(d) Restormer (e) Deflow (f) Ours + DPIR

Figure 11. Full restored DPDD image for each method analyzed

Figure 12. Samples from our parking-lot dataset. All 6 images are at f-stop 5.6

In Fig. 11, we show the full reconstructed images used for Fig. 4. The first 2 images are methods which use the real degradation which are our baselines. The 3 following images come from the other methods we compare ours to and the last image is our method.

B.3. Parking-lot experiment

Dataset We used a Panasonic DC-GX9 camera in aperture priority mode with a Leica DG Summilux 25mm f/1.4 II lens to take 22 distinct photographs in a parking lot at apertures f/16 and f/5.6. We used a tripod for stabilisation but the two images are still not perfectly aligned (for this reason we do not provide any quantitative metric on experiments using this dataset). We process the raw images using dcraw [13] to fix white balance, image highlights, demosaicking, and a small amount of noise reduction. We then converted images to png format. A low-resolution sample of the images in the dataset is shown in Fig. 12. The high-resolution counterparts have 5200x3904 pixels.

Training We started from a scratch CFM model as the size of the one for DDPD experiment, we train it on 64-sized patches from the 18 parking lot images at f-stop 16. For the distribution-matching step we used the the central 768x768 pixels of f/5.6 images as the clean distribution and learn a 8x8 grid of 13x13 RGB kernels. We compare predicted kernels with Eboli et al. [16] in Figs. 13 and 14. Our PSFs are consistent across images by design, while the competing method adapts between different images. The PSFs predicted by our method more accurately recover the true blur, as seen in Figs. 6 and 15.

Hyperparameters The first step consists in learning a diffusion model on a single, low-resolution image. We used a flow-matching model with about 4M parameters, trained on patches of size 64x64. We used a batch-size of 1024 patches, without any added noise, learning rate of 0.01 and 0.05 as an EMA value. We trained for 16M patches. To learn a different blur filter for different locations in the image plane, we condition the diffusion model on the original patch's location in the full image. We do this by concatenating the input image with two positioning-channels (for the x and y axes). Since the dataset is small, to avoid completely overfitting the diffusion model to the location-specific conditioning, we replace the conditioning for 20% of the patches with a randomly chosen location. For the second step we used a smaller batch size of 128, learning rate of 0.00001 for the auxiliary diffusion model and 0.00004 for the kernel network (described in the next paragraph). We added sparsity regularization (l1) with strength 0.01 and gaussian prior regularization with strength 0.1. The second step was trained for 6M patches. Here we wished to learn a location-conditioned degradation which, when applied to any part of the central portion of a clean image, it would replicate the degradations of a specific location of the corrupted images. To achieve this, we conditioned the step-2 model (whose inputs are patches from the central portion of clean images) on random patch locations (spanning the full image size). Like for DPDD we parameterized the degradations by a 8x8 grid of kernels, which unlike DPDD are different for the three RGB channels (to represent chromatic aberrations accurately). Final filters for reconstruction (which is patch-wise) are linearly interpolated from this 8x8 grid. The learned PSF grid is shown in Fig. 13. We additionally added a trainable parameter to learn the noise standard deviation of the images, which was assumed to be fixed throughout the image. The noise estimate was low (around 0.003) because similar amounts of noise were present in the clean and corrupted images. For the third step, we used plug-and-play method DPIR [57] which uses a deep natural image prior. We found the results were very similar to classical Wiener deconvolution, apart for some extra noise removal with DPIR. Additional qualitative reconstructions are shown in Fig. 15.

B.4. Super Resolution

Hyperparameters The first step consists in learning a diffusion model on a single, low-resolution image. We used a small flow-matching model with about 4M parameters, trained on patches of size 32x32. We used a batch-size of 512 patches, without any added noise, learning rate of 0.01 and a small EMA value. We trained for 1M images (or approximately 2000 steps). For the second step we used a smaller batch size of 64, learning rate of 0.00001 for the auxiliary diffusion model and 0.00008 for the kernel network (described in the next paragraph). We added center-regularization with strength 0.01, sparsity regularization (l1) with strength 0.1 and sum-to-one regularization with strength 0.1. Note that the kernel network was not forced to output normalized kernels (unlike all previous experiments), so the sum-to-one regularization was important. The second step was trained for 512k images (or approximately 8000 steps).

Kernel parameterization Similarly to KernelGAN [6] and DCLS [33] we use a linear convolutional network to model the blur kernel. This network can be used directly as forward model $\mathcal{A}_\omega$, and can be collapsed into an explicit kernel by convolving a dirac function (i.e. an image with a single non-zero pixel in the center). We could confirm the findings of Bell-Kligler et al. [6] that this parameterization is easier to train than an equivalent CNN with a single layer. We used 4 layers with kernel-sizes 7, 5, 5 and 1, (for an overall receptive field of 15) no bias and 64 channels.

Additional results In Fig. 16 we show zoom-ins on four different images (35, 36, 37 and 39) from the DIV2KRK dataset as reconstructed with different methods, as well as the blur kernels predicted by the respective methods. We can see that while DANv2 [35] has better reconstruction, the predicted kernels do not show significant differences to the ones predicted with our method. The better reconstruction is due to the relatively lower accuracy of ZSSR compared to the integrated super-resolver in DANv2. FKP has decent reconstruction because it always predicts blur kernels which are close to being isotropic Gaussians with a small standard deviation, while DKP predicts kernels with a higher variance which are not very close to the true ones.

In Fig. 17 we plot the distribution of the difference in PSNR between blind SR methods and their non-blind counterpart. Note that this can be greater than zero when due to random stochasticity of the non-blind method, the estimated kernel results in a better reconstruction than the true one.

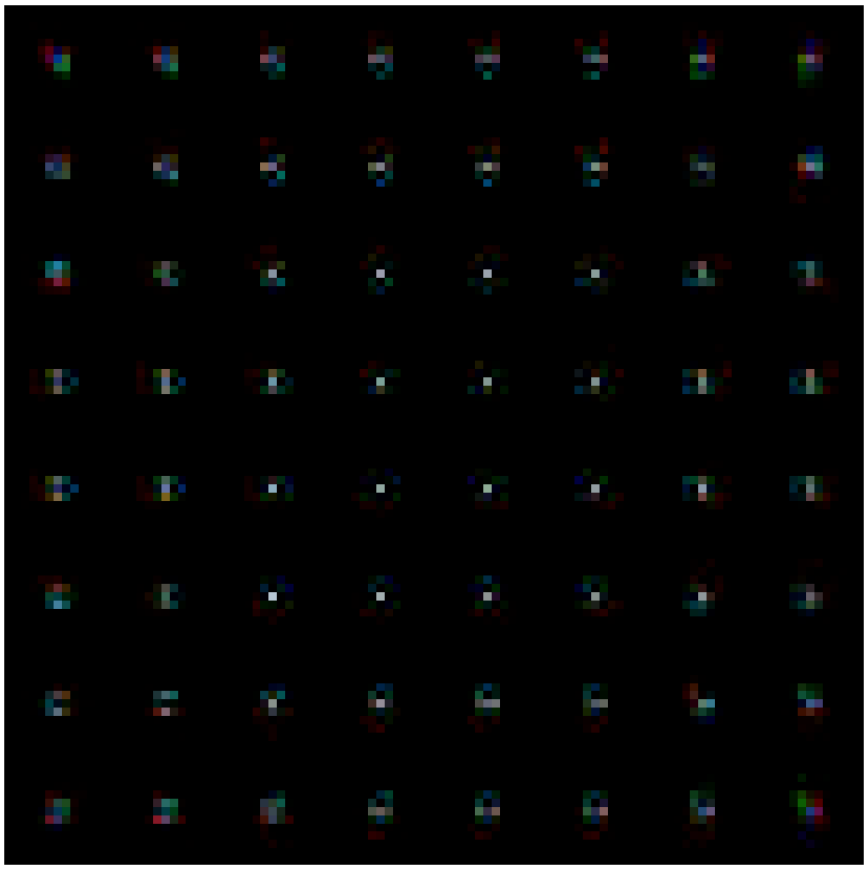


Figure 13. Lens PSF predicted by our model brightened x2. Note how the chromatic aberration (different position of red and blue channels) varies smoothly between the top, bottom and left, right parts of the lens.

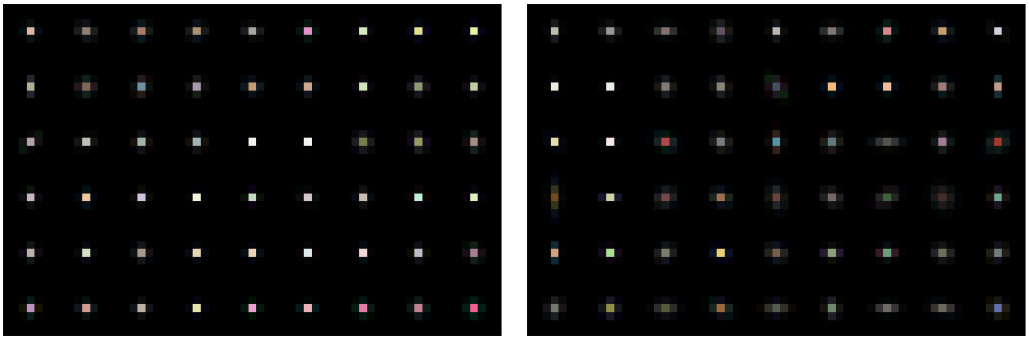


Figure 14. Lens PSF by Eboli [15] for two different images. Note that this method corrects for color aberration in a separate successive step. The PSFs here are not brightened, and hence much smaller than those in Fig. 13.

Metrics We specify here the parameters used to calculate metrics in Tab. 4. Firstly, we found USRNet to produce very strong border artifacts which – if included – would have completely skewed the results. For this reason we crop 60 pixels off each edge for every image before computing metrics. For the PSNR, in order to maintain consistency with results reported in the literature, we used the Y-PSNR (i.e. computed on just the luminance channel, after converting RGB images to YCbCr). For the LPIPS metric we used the "alex-net" network. The PSNR on kernels was computed after padding the kernels with zeros up to size 25x25 and after shifting them so they were appropriately centered. The NCC metric is the 2d normalized cross-correlation, which can better account for small centering errors in the kernels.

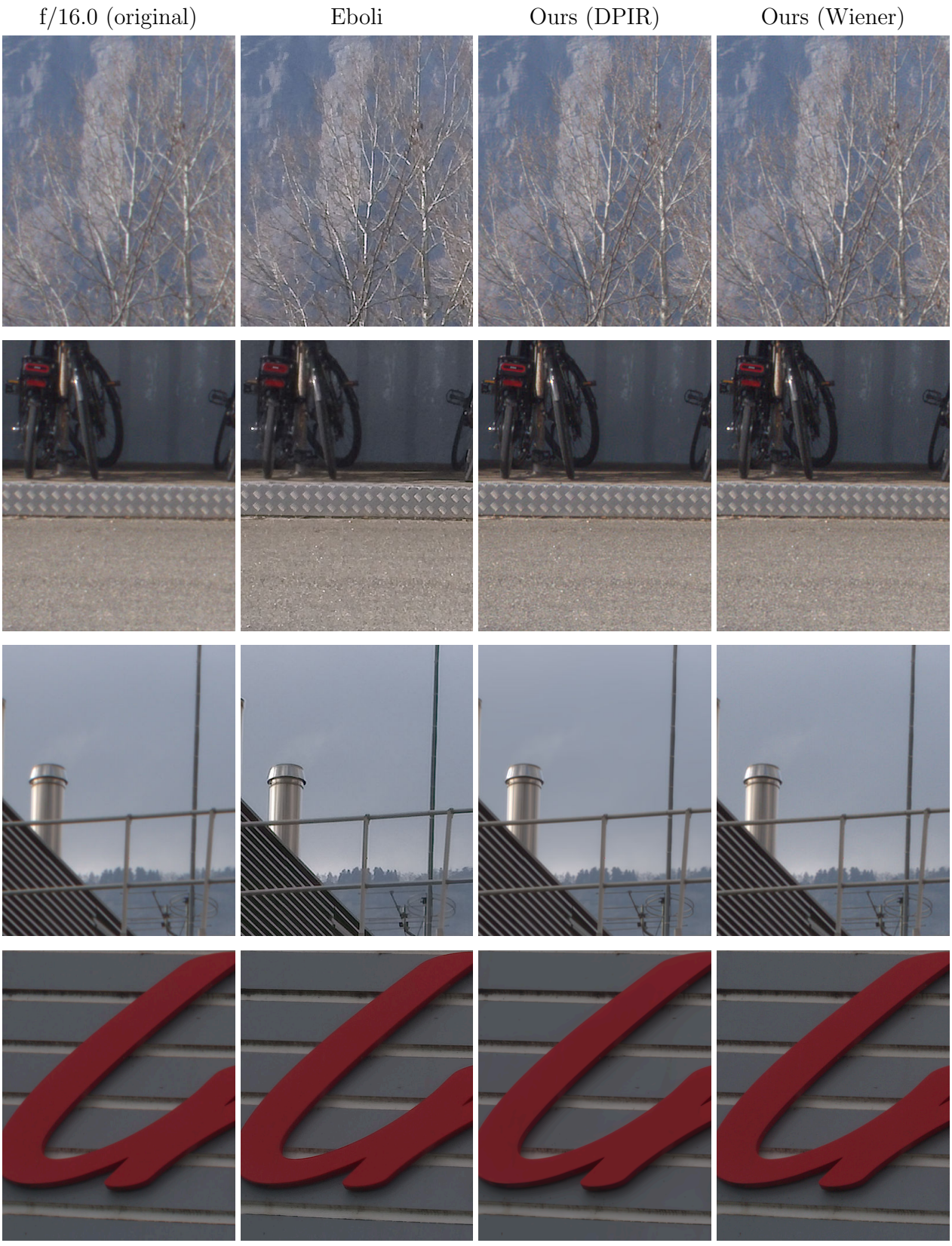


Figure 15. Additional reconstructions on parking lot data (best viewed zoomed-in).

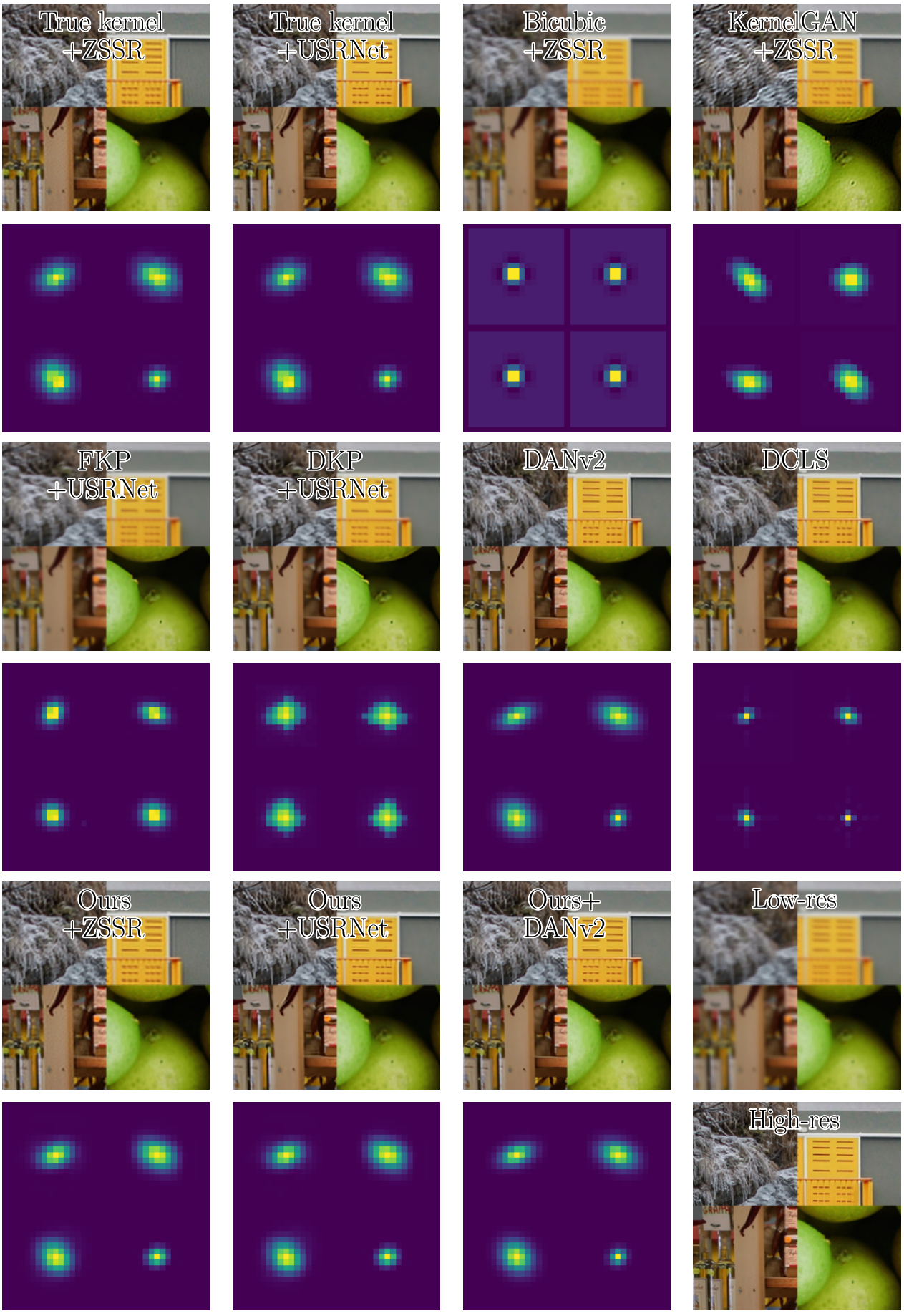


Figure 16. Super-resolution: qualitative results. Note that DCLS kernels are not comparable as they live in a different space.

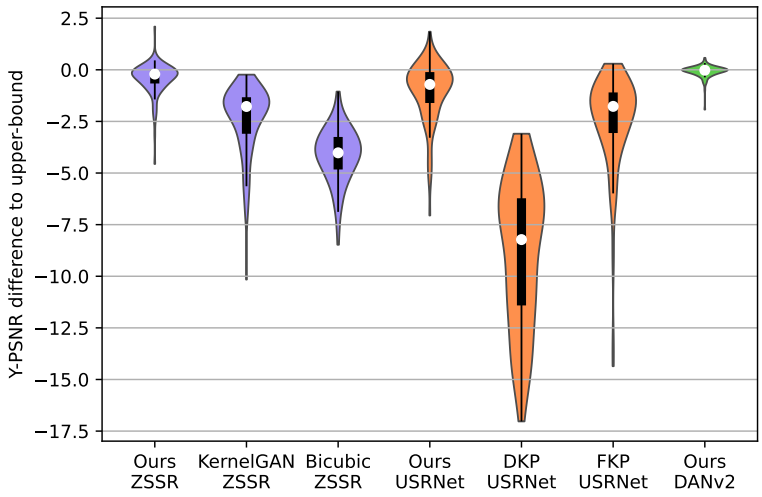


Figure 17. Super-resolution: Performance gap of two-step methods compared to their natural upper-bound. The metric considered is the Y-PSNR (PSNR on the luminance only).