Polyblur: Removing mild blur by polynomial reblurring Supplementary Material

Anonymous CVPR 2021 submission

Paper ID 4417

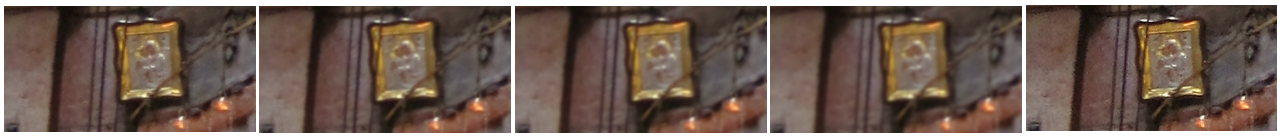


Figure 1: Polyblur removes mild blur by combining multiple applications of the estimated blur.

The core idea in our polynomial deblurring algorithm (polyblur) is to combine multiple applications of the estimated blur kernel in a way that allows us to approximate the inverse. An example of this is shown in Figure 1.

1. Blur model validation and calibration

The Gaussian blur estimation presented in Section 4 is based on computing gradient features. The gradient feature f_θ is defined as,

$$f_\theta = \min_{\psi\in[0,\pi)} f_\psi = \min_{\psi\in[0,\pi)} \max_{\mathbf{x}} |\nabla_\psi v(\mathbf{x})|, \qquad (1)$$

where $v(\mathbf{x})$ is the input image, and $\nabla_\psi f(\mathbf{x})$ is the directional derivative of v at direction ψ. Then, the strength of the blur is computed by,

$$\sigma_0 = \sqrt{\frac{c^2}{f_\theta^2} - \sigma_b^2}, \quad \sigma_1 = \sqrt{\frac{c^2}{f_{\theta_\perp}^2} - \sigma_b^2}. \qquad (2)$$

To calibrate c and σ_b, we proceed as follow. Given a set of 50 sharp high quality images, we simulate $K = 1000$ random Gaussian blurry images, by randomly sampling the blur space and the image set. The Gaussian blur kernels are generated by sampling random values for $\sigma_0 \in [0.3, 4]$ and $\rho \in [0.15, 1]$. Additive Gaussian white noise of standard deviation 1% is added to each simulated blurry image.

Examples of simulated Gaussian blur kernels are shown in Figure 2. For each of the blurry images we compute the gradient features according to (1). The parameters c and σ_b are estimated by minimizing the mean absolute error (MAE). The calibrated parameters are $c = 89.8$ and

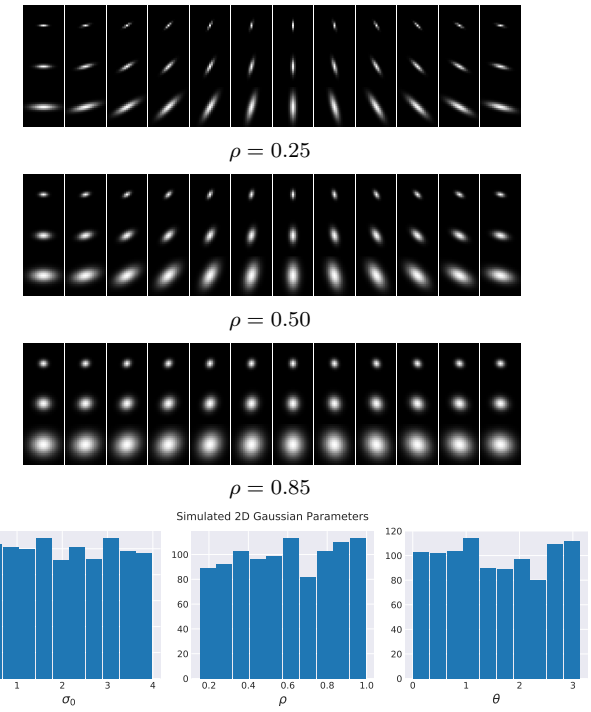


Figure 2: Gaussian Blur kernel examples and distribution of simulated parameters.

$\sigma_b = 0.764$. Note that the values of c and σ_b are implementation dependent (e.g., the finite difference scheme used to compute image gradient).

Figure 3-left shows the relation between the inverse of the estimated gradient feature (i.e., $1/f_\theta$) and the simulated blur kernel σ_0 value. Each of the blur points represents one

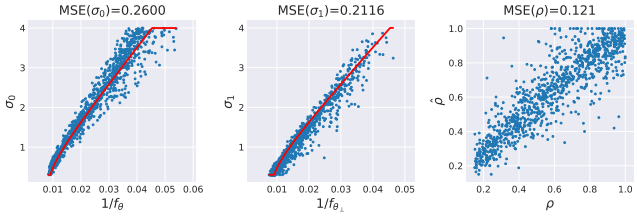


Figure 3: Gaussian blur model calibration. Error in the estimation of σ_0 and σ_1 (and ρ).

simulated image. As we can see, sharp images that have very low blur values ($\sigma_0 \ll 1$) have very similar feature values. The same analysis holds for the gradient feature at the orthogonal direction, and its relation to $\sigma_1 = \rho\sigma_0$ (Figure 3-middle). This validates our *Assumption 1*:

Assumption 1. *In a sharp image, the maximum value of the image gradient in any direction is mostly constant and roughly independent of the image.*

Additionally, Figure 3 shows the almost linear relation (with some spread) between the inverse of the gradient feature $1/f_\theta$ and the blur level σ_0. This is exactly *Assumption 2*:

Assumption 2. *If a sharp image is affected by Gaussian blur, the blur level in the direction of the principal axis will be linearly related to the inverse of the maximum image gradient in the principal directions.*

Our model contemplates for slight blur that the gradient operator may have introduced when computing the gradient features (σ_b). This is further analyzed at the end of this section for a synthetic image.

In Figure 3-right we show a plot of the real ρ value of the simulated blur kernel and the estimated one. Although there are some outliers, the estimation is in general close to the real value (MSE(ρ) = 0.121).

Estimation of blur direction θ. The estimation of the blur direction is done by computing the angle θ with minimum gradient feature value in Eq. (1). Figure 4 shows the error on the estimation of the angle for each simulated blur. As we can see, the error in the angle is quite low for blur kernels highly directional ($\rho < 0.5$). For large values of ρ (e.g., $\rho \in [0.75, 1.0]$, the kernels are almost isotropic, and the estimation is inaccurate. Nevertheless, being almost isotropic, the kernel shape is not affected by the angle value in this case.

Error metrics on estimated kernel values. The ultimate goal in blur estimation is to estimate a blur that is close to the real one. In Figure 5 we present the distribution of two different error metrics that evaluate the distance between the estimated kernel and the simulated one directly on the

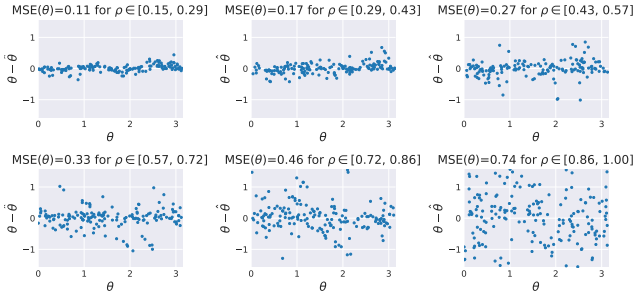


Figure 4: Blur model calibration. Error in estimated parameters. Calibrated parameters are $c = 89.8, \sigma_b = 0.764$.

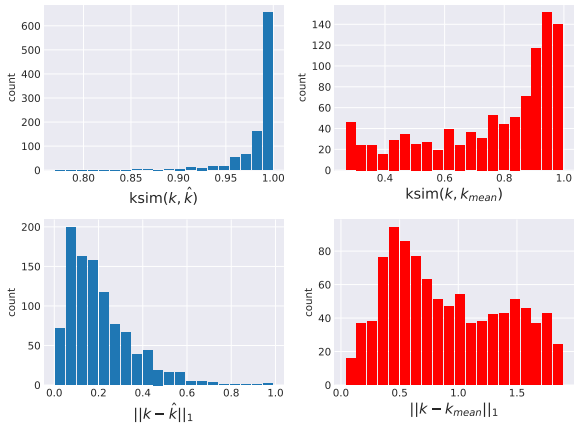


Figure 5: Kernel error metrics on the simulated data. Top-row shows the histogram of the kernel similarity between estimated kernels and the respective ground-truth one (left), and between the ground-truth kernels and a fixed isotropic Gaussian kernel (denoted by k_{mean}) having standard deviation the mid-value on the simulated range (right). Bottom-row shows the histogram of the ℓ_1 difference between the estimated kernels and the respective ground-truth one (left), and between the ground-truth kernels and k_{mean}.

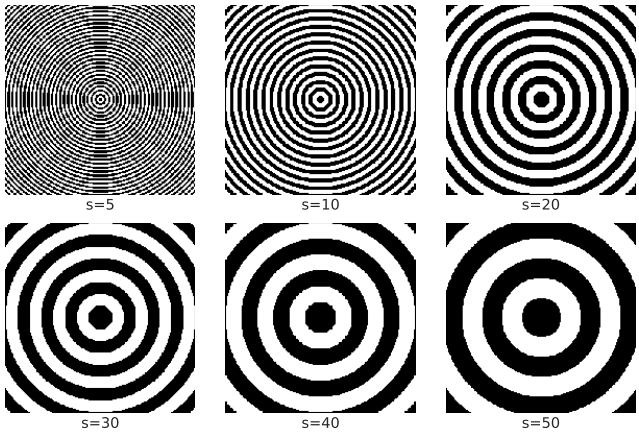
kernel space. The first metric is the kernel similarity, which is the normalized cross correlation:

$$\text{ksim}(k, \hat{k}) = \frac{1}{\|k\|\|\hat{k}\|} \sum_i k_i \hat{k}_i. \tag{3}$$

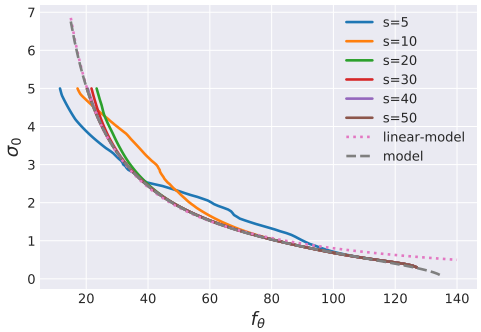
When both kernels are equal the kernel similarity is 1. The second metric we compute is the ℓ_1 norm between estimated kernel values. To give an idea of the range of both metrics, we also present the error distribution of the real kernel and an average kernel[1]. This shows the estimations is accurate.

Concentric circles: A synthetic example. We generated a family of synthetic images with concentric circles at different distances (controlled by a parameter $s \in [5, 50]$, see

[1]The average kernel is defined as the isotropic kernel having standard deviation the central value of the simulated range [0.35, 4.0].



(a) Concentric circles pattern.



(b) Gradient feature f_θ as a function of simulated blur level σ_0.

Figure 6: Blur model on a synthetic image.

Figure 6a and blurred them with isotropic Gaussian blur of intensity ($\sigma_0 \in [0.3, 5]$). In Figure 6b we show the computed gradient feature (f_θ) for the the different blurry images (different σ_0) on each simulated pattern ($s = 5$ to $s = 50$). We calibrated the blur model (i.e., estimated c and σ_b values) using the $s = 50$ synthetic pattern image. The model is highly accurate when the circular rings are separated enough so the image can be locally considered a step-edge ($s \geq 20$). However, when the concentric circles are very close (e.g., $s = 5$), the model is not accurate and the blur estimation is biased.

For comparison purposes we also fit a purely linear model that directly maps the inverse of the gradient feature to the sigma value. The linear model (which is the proposed model with $\sigma_b = 0$) is very close to the proposed model except in low sigma values. This is because the proposed model takes into account the (very little) blur σ_b that the gradient operator introduces, leading to a more precise estimation.

2. Blur estimation implementation details

In Algorithm 1 we present the pseudo-code for our estimation of Gaussian blur. We follow Section 4 to estimate the blur. As the first step the input image is normalized using quantiles ($q = 0.0001$ and $1 - q$) to be robust to outliers. From the image gradient (u_x, u_y) we compute the directional derivative at n_{angles} (usually $n_{\text{angles}} = 6$) uniformly covering $[0, \pi)$. The maximum direction of the magnitude for each sample angle is found. Among the n_{angles} maximum values, we find the minimum value and angle (f_0, θ_0) through bicubic interpolation. Using Eq. 2 we compute σ_0 and σ_1 (and compute $\rho = \sigma_1/\sigma_0$).

Algorithm 1: Gaussian blur estimation.

input : image u, q, n_{angles}, $\sigma_{\max}$, $\sigma_{\min}$, $\rho_{\max}$, $\rho_{\min}$
output: Blur parameters: σ_0, ρ, θ.

// 1. Compute gradient features;
n = Normalize(u, q);
u_x,u_y = ComputeGradient$(n, \sigma_{\max})$;
for $i \in [0, n_{angles}]$ **do**
  $\Psi_i = i \cdot \pi / n_{\text{angles}}$;
  $u_{\Psi_i} = u_x \cdot cos(\Psi_i) - u_y \cdot sin(\Psi_i)$;
  f_{Ψ_i} = Max$(|u_{\Psi_i}|)$;
end
I_Ψ = Interpolate$(\{f_\Psi\})$;
f_θ, θ_0 = Min(I_Ψ);
$f_{\theta_\perp}, \theta_{0_\perp}$ = Ortho(f_θ, θ_0);

// 2. Compute and clamp Gaussian;
σ_0, σ_1 = Regression$(f_\theta\ , f_{\theta_\perp})$ // From Eq. (2);
Clamp$(\sigma_0, \sigma_1, \sigma_{\max}, \sigma_{\min})$;
$\rho = \sigma_0/\sigma_1$;
Clamp$(\rho, \rho_{\min}, \rho_{\max})$;

The gradient features can be efficiently computed in parallel not just between different angles but also between each different pixel. Computing the maximum can be represented as a gather operation that can be optimized using shared memory and tiling. Within our pipeline this represents around $40 - 53\%$ of the computation, this comes from the fact that we compute a maximum magnitude n_{angles} times.

3. Comparison on DIV2K dataset

In figures 7–11 we present additional results to the one presented on Section 6. We generated a mild-blur dataset by artificially blurring sharp images from the DIV2K dataset [1]. Each of the 100 images in the validations dataset were blurred by a random Gaussian blur kernel of different sizes, shapes, and orientations (i.e., $\sigma_0 \sim \mathcal{U}[0.3, 4]$, $\rho \sim \mathcal{U}[0.15, 1.0]$, $\theta \sim \mathcal{U}[0, \pi]$). Additive white Gaussian noise of standard deviation 1% was added on top. We compared Polyblur (one, two, and three iterations) to the following adaptive sharpening or deblurring methods: SRN-Deblur [9], Sparse Deblurring [10], DeblurGANv2

(inception and mobilenet architectures) [6], Spectral Irregularities [3], L0-Deblur [8], GLAS [11], Guided Filter [4], Convolutional Deblurring [5].

4. Deblurring before super-resolution

Figures 12 and 13 present additional results to Section 6 on deblurring before super-resolution. We apply Polyblur as a pre-step before using an off-the-shelf deep network for doing $4\times$ image upscaling. We trained from scratch an EDSR [7] network with 32 layers and 64 filters using DIV2K training dataset. Polyblur produces the best quantitative and qualitative results. The evaluation is done on the DIV2KRK dataset introduced in [2].

5. Comparison on images *in the wild*.

In figures 14 to 18 we present additional results of Polyblur applied to some images in the wild (Section 6). We compare Polyblur (one, two and three iterations) to the following adaptive sharpening or deblurring methods: SRN-Deblur [9], Sparse Deblurring [10], DeblurGANv2 (inception and mobilenet architectures) [6], Spectral Irregularities [3], L0-Deblur [8], GLAS [11], Guided Filter [4], Convolutional Deblurring [5]. Polyblur manages to remove mild blur, as the one present in most images, without introducing any new artifacts.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017. 3

[2] Sefi Bell-Kligler, Assaf Shocher, and Michal Irani. Blind super-resolution kernel estimation using an internal-gan. In *Advances in Neural Information Processing Systems*, pages 284–293, 2019. 4, 10, 11

[3] Amit Goldstein and Raanan Fattal. Blur-kernel estimation from spectral irregularities. In *European Conference on Computer Vision*, pages 622–635. Springer, 2012. 4

[4] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE transactions on pattern analysis and machine intelligence*, 35(6):1397–1409, 2012. 4

[5] Mahdi S Hosseini and Konstantinos N Plataniotis. Convolutional deblurring for natural imaging. *IEEE Transactions on Image Processing*, 29:250–264, 2019. 4

[6] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8878–8887, 2019. 4

[7] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017. 4

[8] Jinshan Pan, Zhe Hu, Zhixun Su, and Ming-Hsuan Yang. l_0-regularized intensity and gradient prior for deblurring text images and beyond. *IEEE transactions on pattern analysis and machine intelligence*, 39(2):342–355, 2016. 4

[9] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8174–8182, 2018. 3, 4

[10] Haichao Zhang, David Wipf, and Yanning Zhang. Multi-image blind deblurring using a coupled adaptive sparse prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1051–1058, 2013. 3, 4

[11] Xiang Zhu and Peyman Milanfar. Restoration for weakly blurred and strongly noisy images. In *2011 IEEE Workshop on Applications of Computer Vision (WACV)*, pages 103–109. IEEE, 2011. 4

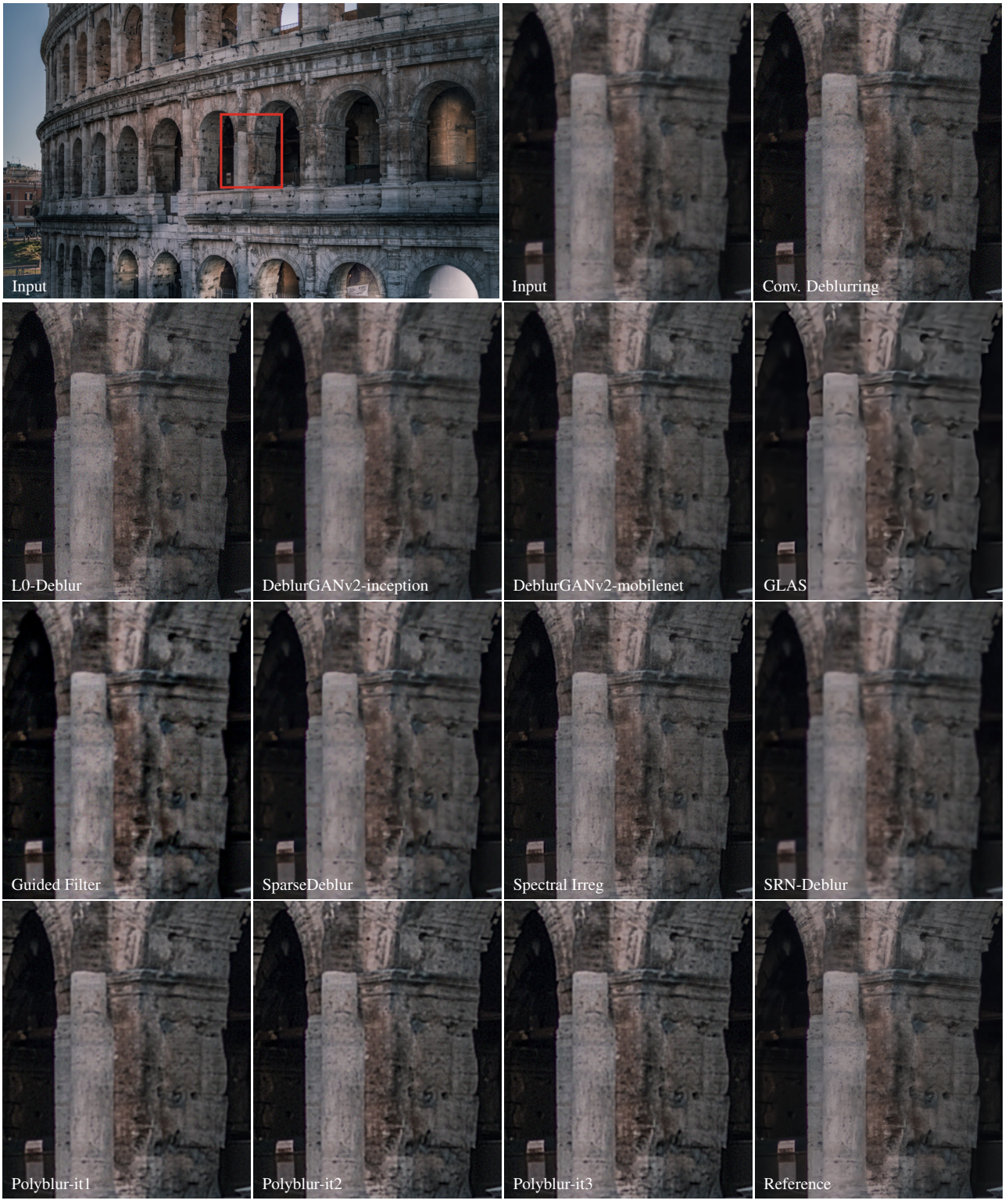


Figure 7: Example of comparison on one image from the DIV2K validation dataset with synthetic blur.

Figure 8: Example of comparison on one image from the DIV2K validation dataset with synthetic blur.

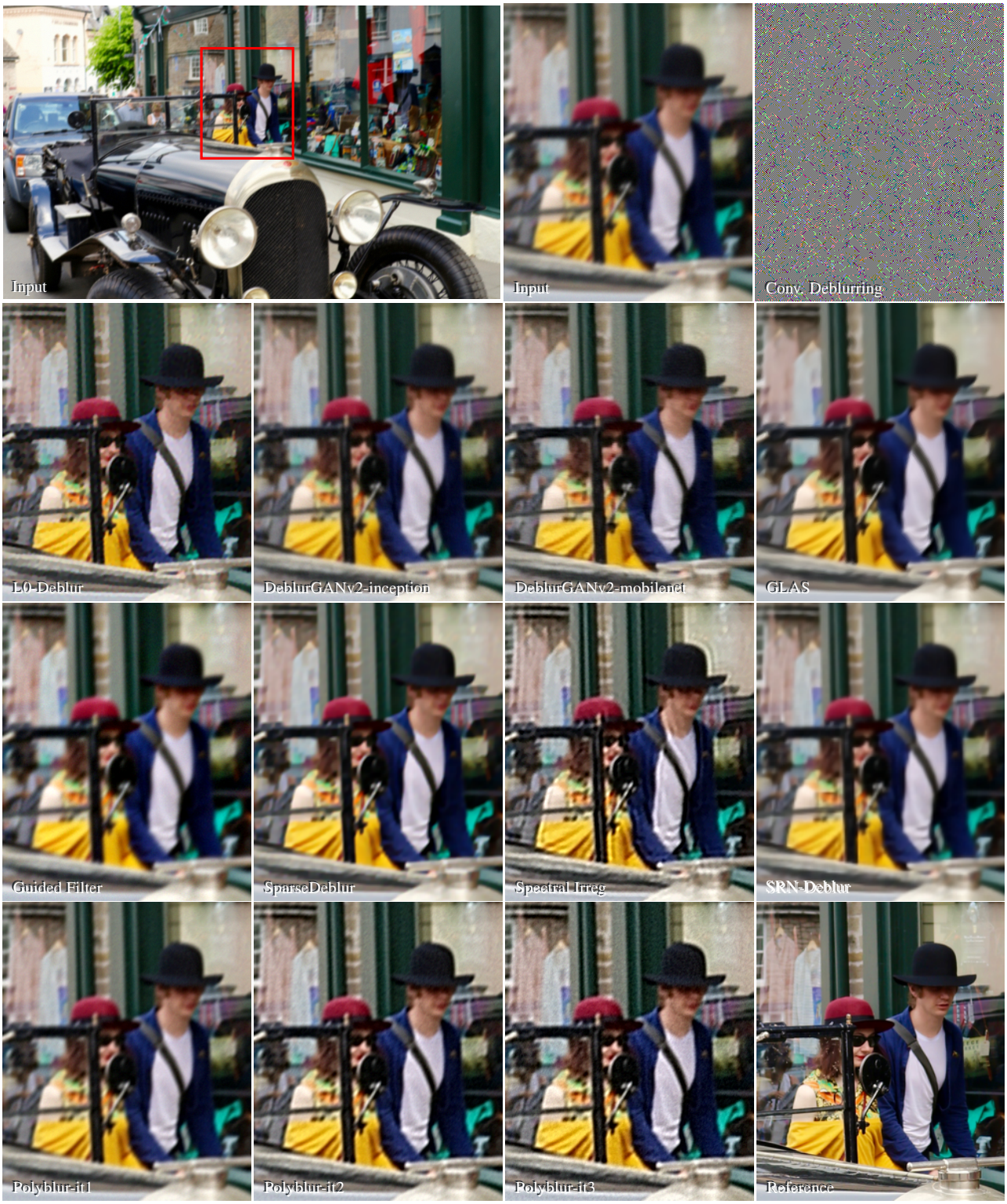


Figure 9: Example of comparison on one image from the DIV2K validation dataset with synthetic blur.

Figure 10: Example of comparison on one image from the DIV2K validation dataset with synthetic blur.

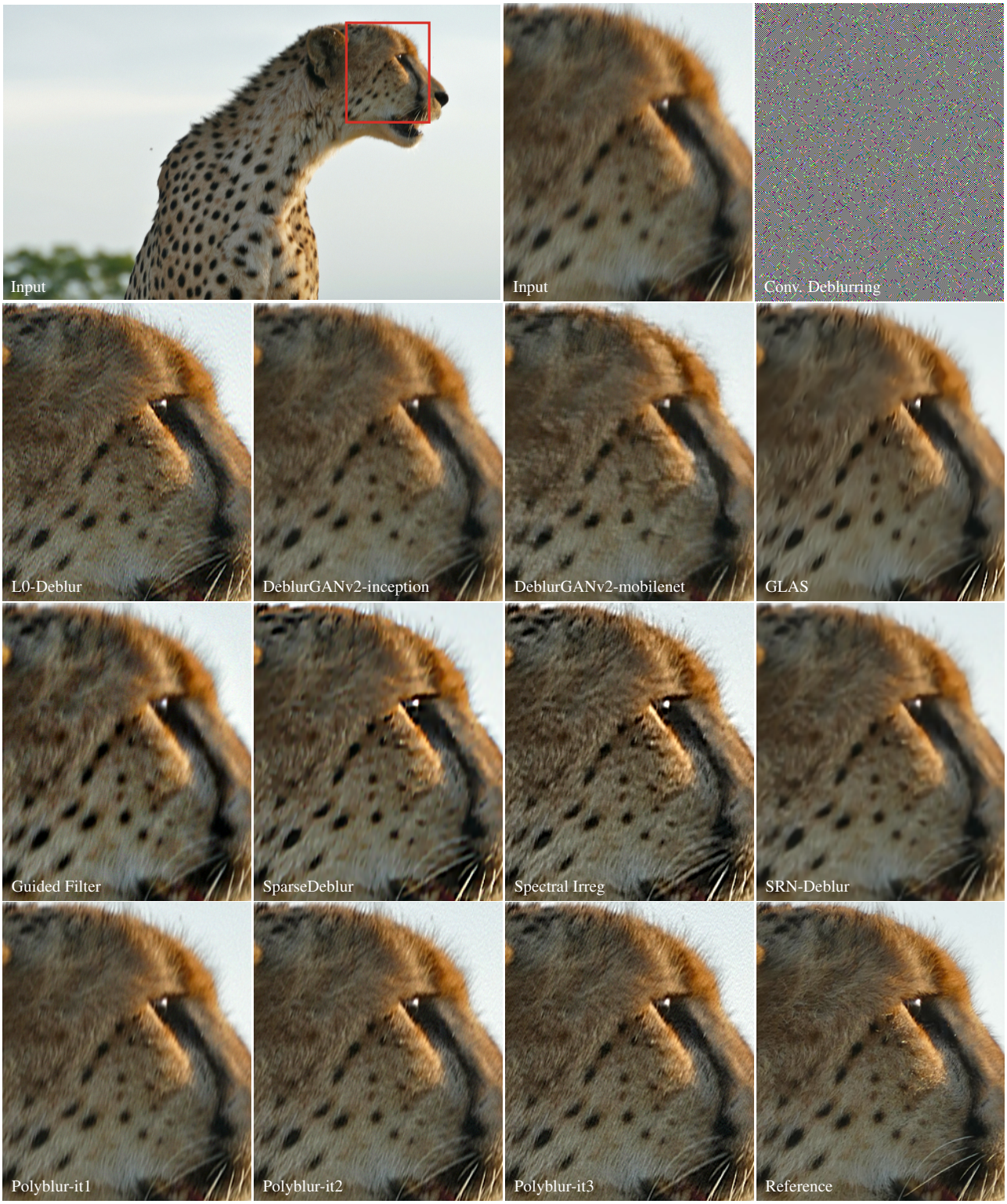


Figure 11: Example of comparison on one image from the DIV2K validation dataset with synthetic blur.

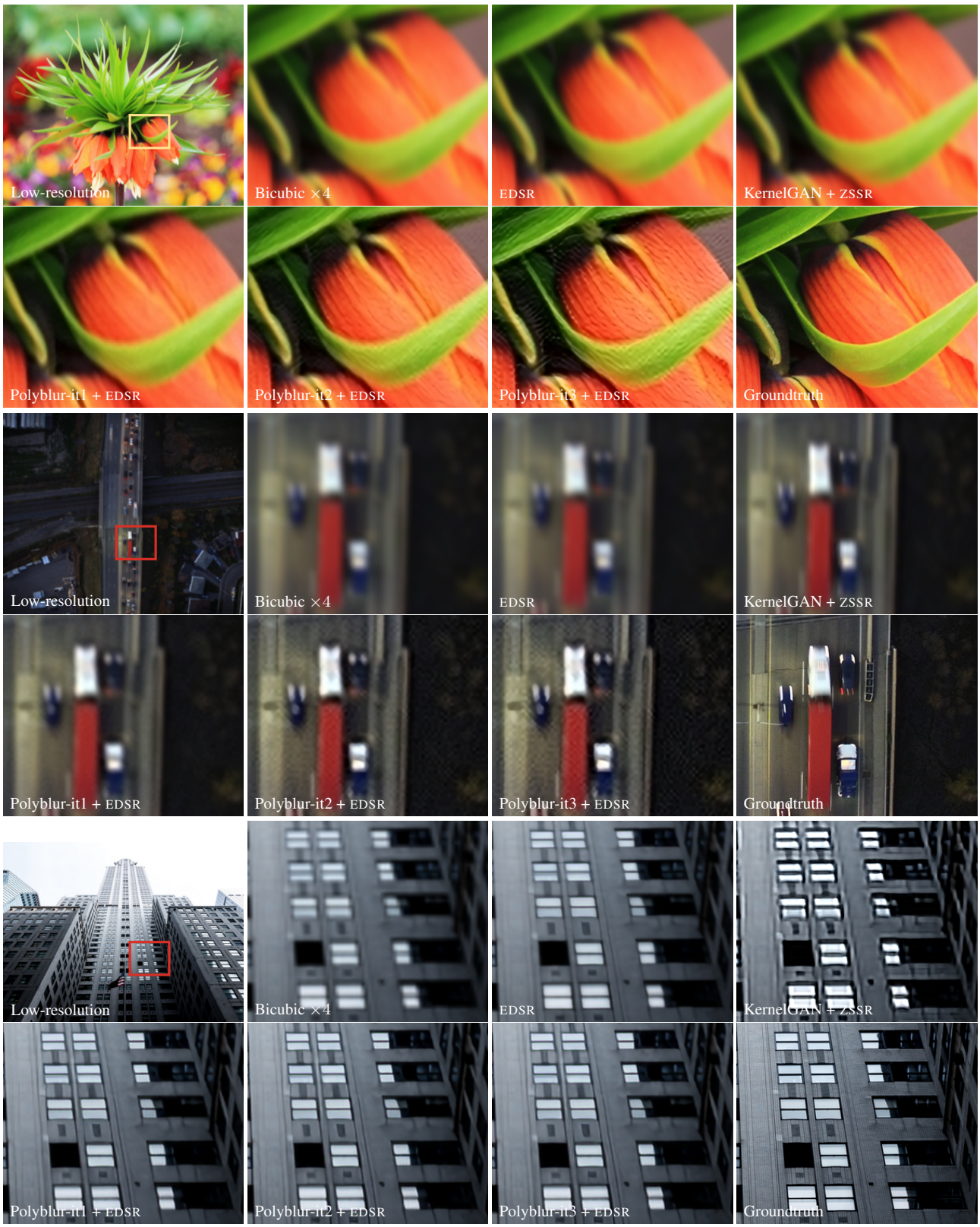


Figure 12: DIV2KRK [2] 4× upscaling with *unknown* kernel and out-of-the-shelf super-resolution model.

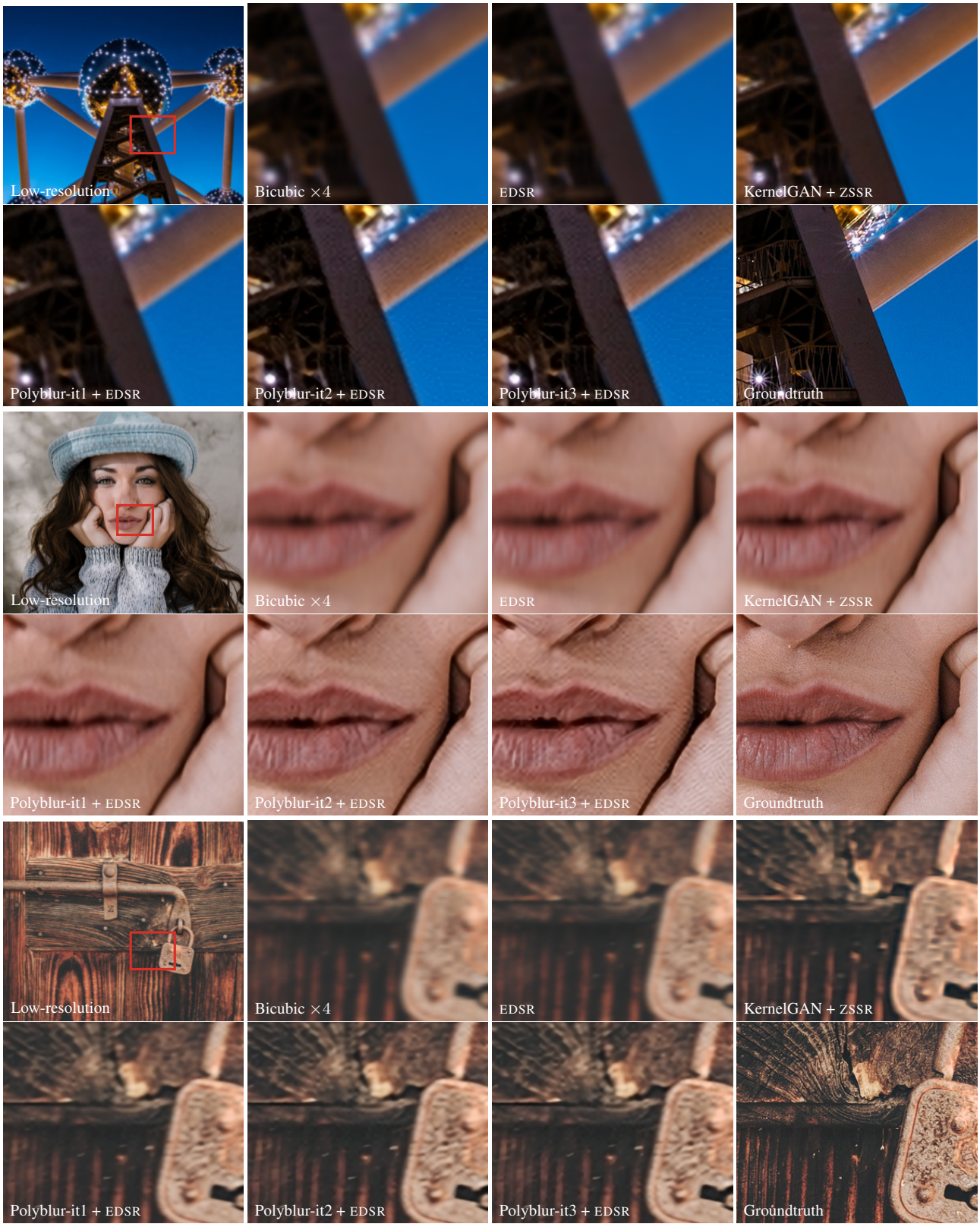


Figure 13: DIV2KRK [2] 4× upscaling with *unknown* kernel and out-of-the-shelf super-resolution model.

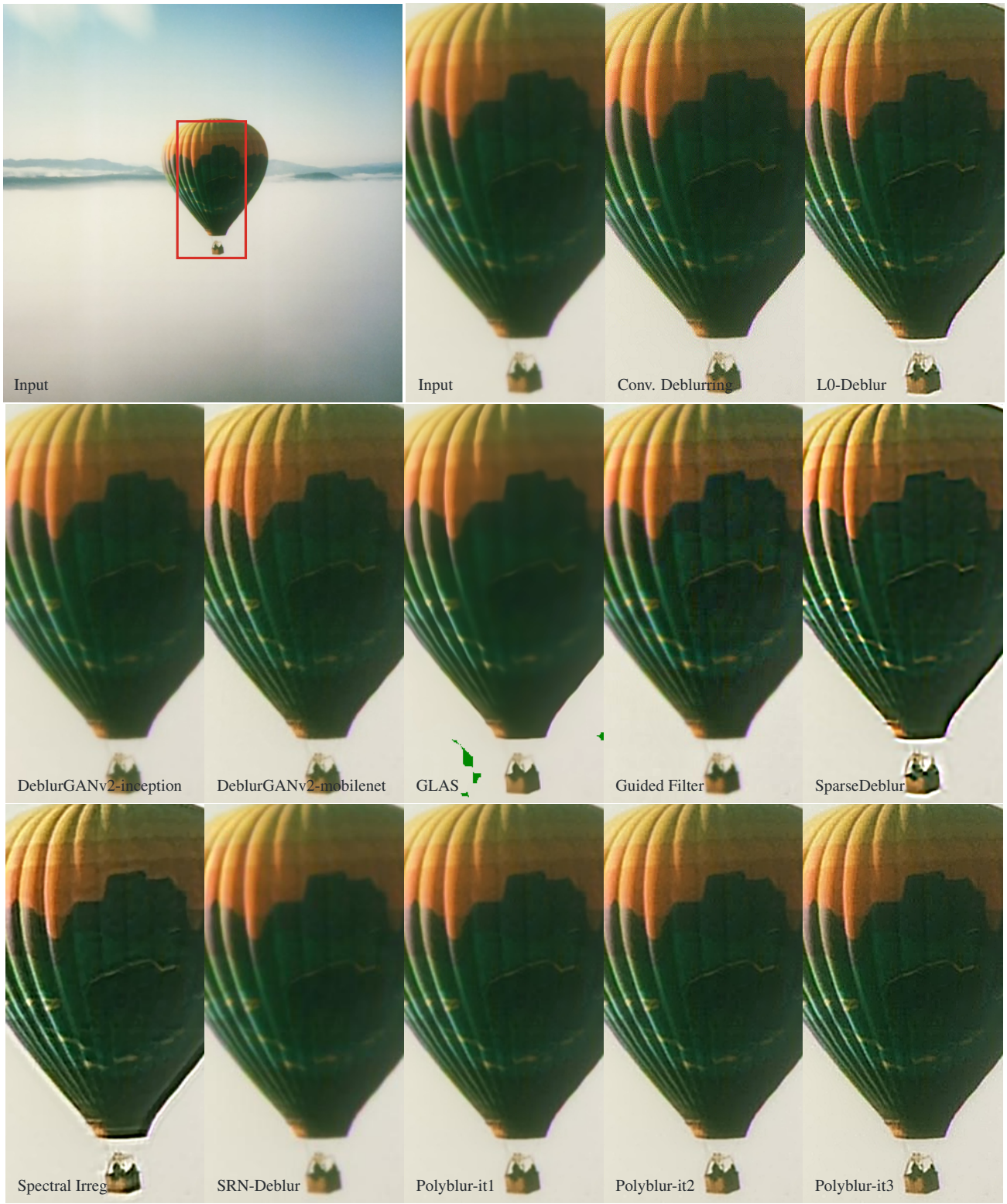


Figure 14: Example of removing mild blur *in the wild*.

Figure 15: Example of removing mild blur *in the wild*.

Figure 16: Example of removing mild blur *in the wild*.

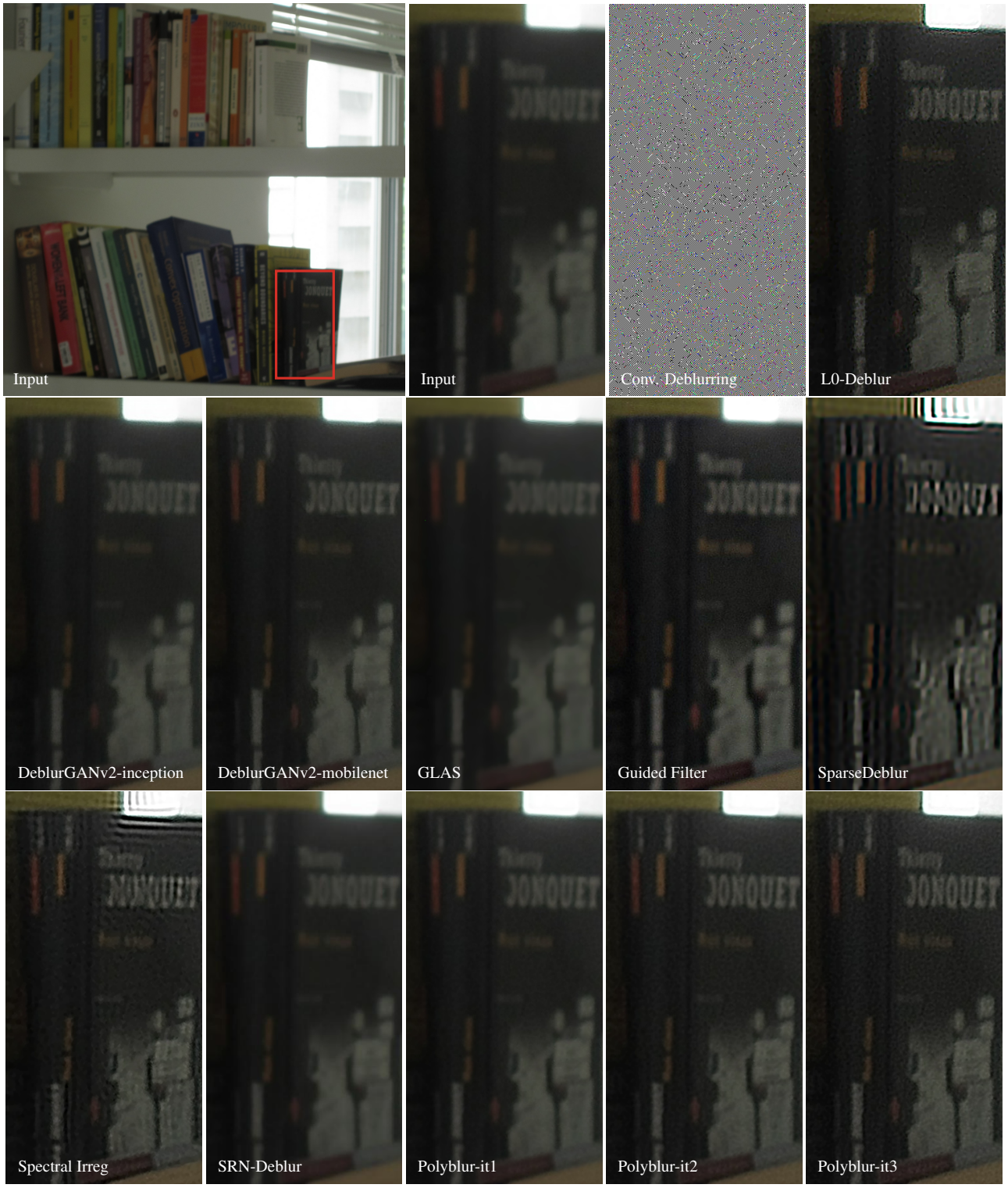


Figure 17: Example of removing mild blur *in the wild*.

Figure 18: Example of removing mild blur *in the wild*.