Deep Reparametrization of Multi-Frame Super-Resolution and Denoising

Supplementary Material

Goutam Bhat Martin Danelljan Fisher Yu Luc Van Gool Radu Timofte
Computer Vision Lab, ETH Zurich, Switzerland

We provide additional details and analysis of our approach in this supplementary material. Section 1 provides a derivation of the closed-form expressions for the steepest-descent steps (8) from the main paper. The linearity of the warping operator is discussed in Section 2. Our entire inference pipeline is detailed in Section 3. The network architectures employed for the RAW burst super-resolution and burst denoising tasks are described in detail in Section 4. An analysis of our certainty predictor W is provided in Section 5. Section 6 contains an additional ablative analysis of our approach. Further qualitative comparison on the burst super-resolution and denoising datasets are provided in Section 7.

1. Derivation of Steepest-Descent Steps

In this section, we derive the closed-form expressions for the gradient $g^j = \nabla L(z^j)$ of loss (7) in the main paper, as well as the steepest-descent step lengths α^j. Our optimization objective (7) from the main paper is here restated as,

$$L(z) = \sum_{i=1}^{N} \|r_i\|_2^2 + \lambda \|z\|_2^2 \quad \text{where} \quad r_i = v_i \cdot (E(x_i) - G * \phi_{m_i}(z)) \; . \tag{1}$$

In the following derivation, we will interchangeably treat the entities in (1) either as 3D feature maps or as corresponding vectors. By using the chain rule, the gradient of (1) is computed as,

$$g = \nabla L(z) = \sum_{i=1}^{N} 2 \left[\frac{\partial r_i}{\partial z} \right]^{\mathrm{T}} r_i + 2\lambda z \tag{2}$$

$$= \sum_{i=1}^{N} -2 \left[\frac{\partial G * \phi_{m_i}(z)}{\partial z} \right]^{\mathrm{T}} (v_i \cdot r_i) + 2\lambda z \tag{3}$$

$$= \sum_{i=1}^{N} -2 \left[\frac{\partial G * \phi_{m_i}(z)}{\partial \phi_{m_i}(z)} \frac{\partial \phi_{m_i}(z)}{\partial z} \right]^{\mathrm{T}} (v_i \cdot r_i) + 2\lambda z \tag{4}$$

$$= \sum_{i=1}^{N} -2 \left[\frac{\partial \phi_{m_i}(z)}{\partial z} \right]^{\mathrm{T}} \left[\frac{\partial G * \phi_{m_i}(z)}{\partial \phi_{m_i}(z)} \right]^{\mathrm{T}} (v_i \cdot r_i) + 2\lambda z \tag{5}$$

$$= \sum_{i=1}^{N} -2 \phi_{m_i}^{\mathrm{T}} G *^{\mathrm{T}} (v_i \cdot r_i) + 2\lambda z \tag{6}$$

$$= -2 \sum_{i=1}^{N} \phi_{m_i}^{\mathrm{T}} G *^{\mathrm{T}} \left(v_i^2 \cdot (E(x_i) - G * \phi_{m_i}(z)) \right) + 2\lambda z \tag{7}$$

Here, $G*^{\mathrm{T}}$ denotes the transpose of the convolution operator $u \mapsto G * u$, which is the same as the transpose of the Jacobian $\frac{\partial G*u}{\partial u}$. Similarly, $\phi_{m_i}^{\mathrm{T}}$ denotes the transpose of the linear warp operator $z \mapsto \phi_{m_i}(z)$, which corresponds to the transpose of the Jacobian $\frac{\partial \phi_{m_i}(z)}{\partial z}$.

Our step length α^j is computed by performing an optimal line search $\alpha^j = \arg\min_\alpha L(z^j - \alpha g^j)$ in the gradient direction g^j. Since our loss (1) is convex, it has a unique global minima, which is be obtained by solving for the stationary point $\frac{\mathrm{d}L(z^j - \alpha g^j)}{\mathrm{d}\alpha} = 0$. By setting $u = z^j - \alpha g^j$ and applying chain rule, we get

$$0 = \frac{\mathrm{d}L(u)}{\mathrm{d}\alpha} = \left[\frac{\mathrm{d}u}{\mathrm{d}\alpha}\right]^{\mathrm{T}} \nabla L(u) \tag{8}$$

$$= [-g^j]^{\mathrm{T}} \Big(-2 \sum_{i=1}^{N} \phi_{m_i}^{\mathrm{T}} G *^{\mathrm{T}} \left(v_i^2 \cdot (E(x_i) - G * \phi_{m_i}(u)) \right) + 2\lambda u \Big) \tag{9}$$

$$= 2[g^j]^{\mathrm{T}} \sum_{i=1}^{N} \phi_{m_i}^{\mathrm{T}} G *^{\mathrm{T}} \left(v_i^2 \cdot \left(E(x_i) - G * \phi_{m_i} \left(z^j - \alpha g^j \right) \right) \right) + 2\lambda [-g^j]^{\mathrm{T}} (z^j - \alpha g^j) \tag{10}$$

$$= [-g^j]^{\mathrm{T}} g^j + 2 \sum_{i=1}^{N} \alpha [g^j]^{\mathrm{T}} \phi_{m_i}^{\mathrm{T}} G *^{\mathrm{T}} \left(v_i^2 \cdot G * \phi_{m_i} \left(g^j \right) \right) + 2\lambda\alpha [g^j]^{\mathrm{T}} g^j \tag{11}$$

$$= -\|g^j\|^2 + 2\alpha \sum_{i=1}^{N} \| v_i \cdot G * \phi_{m_i} \left(g^j \right) \|^2 + 2\lambda\alpha \|g^j\|^2 \tag{12}$$

In equation (11), we have utilized the closed-form expression (7) for g^j, while also exploiting the linearity of the warp operator ϕ. Using (12), the step length α is obtained as,

$$\alpha^j = \frac{\|g^j\|^2}{\sum_{i=1}^{N} 2\| v_i \cdot G * \phi_{m_i} \left(g^j \right) \|^2 + 2\lambda \|g^j\|^2} \tag{13}$$

2. Linearity of the Warping Operator

In our work, we assume that the warping operation $\phi_m(x)$ is linear. Note that this assumption holds even in the most general case, *i.e.* where the scene motion is given by a pixel-wise optical flow, represented as a pixel-to-pixel mapping $m : \mathbb{R}^2 \to \mathbb{R}^2$. Let $x : \mathbb{R}^2 \to \mathbb{R}^d$ be a continuous representation of a d-dimensional feature map (obtained by *e.g.* bilinear interpolation, which is itself linear). Then the warping operator can be conveniently expressed as a function composition $\phi_m(x) = x \circ m$, *i.e.* $\phi_m(x)(p) = x(m(p))$ for any pixel location $p \in \mathbb{R}^2$. The warping operator ϕ_m is hence linear for any motion m since $\phi_m(ax_1 + bx_2) = (ax_1 + bx_2) \circ m = ax_1 \circ m + bx_2 \circ m = a\phi_m(x_1) + b\phi_m(x_2)$ for any scalars $a, b \in \mathbb{R}$ and feature maps x_1, x_2.

3. Inference Pipeline

Here, we detail the inference pipeline used by our multi-frame image restoration approach. Our approach minimizes the feature space reconstruction loss (7) in the main paper to fuse information from the input images. The entire pipeline is outlined in Algorithm 1. Given the set of input images $\{x_i\}_{i=1}^N$, we first pass each image x_i through the encoder network $e_i = E(x_i)$ to obtain deep image embeddings $\{e_i\}_{i=1}^N$. For each image, we also compute the scene motion m_i w.r.t. to first image x_1, and the certainty values v_i used to weigh our feature space reconstruction loss (Sec. 3.3 in main paper). Next, we estimate the optimal latent encoding $\hat{z}$ of the output image y which minimizes our reconstruction loss $L(z)$. This is achieved by using the iterative steepest-descent algorithm (Sec. 3.4 in main paper). First, we obtain an initial latent encoding z^0 using an initializer network P. The initial encoding is then refined by applying K_{SD} steepest-descent steps (Equation 8 in the main paper) to obtain $z^{K_{\text{SD}}}$. The latent encoding $z^{K_{\text{SD}}}$ is finally passed through the decoder network $\hat{y} = D(K_{\text{SD}})$ to obtain the prediction $\hat{y}$. Note that each step in our inference pipeline is differentiable w.r.t. the parameters of the encoder E, certainty predictor W, initializer P, feature degradation G, and decoder D networks. This allows us to learn each of these modules end-to-end from data, as described in Sec. 3.5 in the main paper.

4. Network Architecture

4.1. RAW Burst Super-Resolution

Here, we provide more details about the network architecture employed for the burst super-resolution task in Section 5.1 and 5.3 in the main paper.

Algorithm 1 Inference pipeline

Input: Multiple images $\{x_i\}_{i=1}^N$, Number of steepest-descent iterations K_{SD}

for $i = 1, \dots, N$ **do**
  $e_i \leftarrow E(x_i)$ *# Map each input image to embedding space*
  $m_i \leftarrow \texttt{MotionEstimator}(x_i, x_1)$ *# Estimate scene motion for each image w.r.t. the first image*
  $v_i \leftarrow W(e_i, e_1, m_i)$ *# Estimate the certainty values for each image*

$z^0 \leftarrow P(e_1))$ *# Obtain initial latent encoding*
for $j = 0, \dots, K_{\text{SD}} - 1$ **do** *# For every steepest-descent iteration*
  $g^j \leftarrow -2 \sum_{i=1}^N \phi_{m_i}^{\text{T}} G *^{\text{T}} \left(v_i^2 \cdot \left(e_i - G * \phi_{m_i}(z^j)\right)\right) + 2\lambda z^j$ *# Obtain gradient of loss (7) in main paper w.r.t. z*
  $\alpha^j \leftarrow \frac{\|g^j\|_2^2}{\sum_{i=1}^N 2\|v_i \cdot \left(G * \phi_{m_i}(g^i)\right)\|_2^2 + 2\lambda \|g^j\|_2^2}$ *# Calculate optimal step-length along gradient direction g^j*
  $z^{j+1} \leftarrow z^j - \alpha^j g^j$ *# Update latent encoding using the estimated step-length α^j*
$\hat{y} \leftarrow D(z^{K_{\text{SD}}})$ *# Decode latent encoding to obtain the output image*

Encoder E: The encoder packs each 2×2 block in the input RAW image along the channel dimension to obtain a 4 channel input. This ensures translation invariance, while also reducing the memory usage. The packed input is passed through a convolution+ReLU block to obtain a 64 dimensional feature map. This feature map is processed by 9 Residual blocks [2], before being passed through a final convolution+ReLU block to obtain a 256 dimensional embedding $E(x_i)$ of the input x_i.

Operator G: We use a convolution layer with stride 2 as our feature degradation operator G. The operator takes a 64 dimensional embedding as input to generate a 256 dimensional output, which is compared with the embedding $E(x_i)$ of the input image to compute the feature space reconstruction error.

Initializer P: We use the sub-pixel convolution layer [5] to generate the initial latent encoding z^0 of the output image $\hat{y}$. The initializer takes the embedding $E(x_1)$ of the first burst image x_1 as input and upsamples it by a factor of 2 via sub-pixel convolution to output z^0.

Certainty Predictor W: The certainty predictor takes the embedding of the input images $\{E(x_j)\}_{j=1}^N$, along with the motion estimation m_i as input. Each image embedding $E(x_i)$ is passed through a convolution+ReLU block to obtain the 64-dimensional feature map e_i. The features e_1 extracted from the reference image x_1 are then warped to i-th image to compute the residual $e_i - \phi_{m_i}(e_1)$, which indicates possible alignment errors. Additionally, we pass the modulo 1 of scene motion m_i mod 1 through a convolution+ReLU block followed by a residual block to obtain the 64-dimensional motion features $\tilde{m}_i$. The encoding e_i, the residual $e_i - \phi_{m_i}(e_1)$, and the motion feature $\tilde{m}_i$ are then concatenated along the channel dimension and projected to 128 dimensional feature map using a convolution+ReLU block. The resulting feature map is passed through 3 residual blocks, followed by a final convolution layer to obtain the output $\tilde{v}_i$. Certainty value v_i for the i-th image is then obtained as the absolute value $|\tilde{v}_i|$ to ensure positive certainties.

Decoder D: The decoder passes the encoding $\hat{z}$ of the output image $\hat{y}$ through a convolution+ReLU block to obtain a 64-dimensional feature map. This feature map is passed through 5 residual blocks, followed by an upsampling by a factor of $s/\tilde{s} = 4$ using the sub-pixel convolution layer [5]. The upsampled 32-dimensional feature map is then passed through 5 additional residual blocks, followed by a final convolution layer which predicts the output RGB image $\hat{y}$.

4.2. Burst Denoising

Here, we provide more details about the network architecture employed for the burst denoising task in Section 5.2 and 5.3 in the main paper.

Encoder E: The encoder concatenates the input image x_i and the per-pixel noise estimate n_i along the channel dimension, and passes it through a convolution+ReLU block. The resulting 32-dimensional output is processed by 4 residual blocks, followed by a convolution+ReLU block to obtain a 64-dimensional encoding $E(x_i, n_i)$ of the input image x_i.

Operator G: We use a convolution layer as our feature degradation operator G. The operator takes a 16 dimensional encoding z of the output image y as input to generate a 64 dimensional output, which is compared with the embedding $E(x_i, n_i)$ of the input image to compute the feature space reconstruction error.

Initializer P: We pass the embedding $E(x_1)$ of the first burst image x_1 through a convolution layer to obtain the initial output image encoding z^0.

Certainty Predictor W: The certainty predictor takes the embedding of the input images $\{E(x_j, n_j)\}_{j=1}^{N}$, the noise estimate n_i, along with the motion estimation m_i as input. Each image embedding $E(x_i, n_i)$ is passed through a convolution+ReLU block to obtain the 16-dimensional feature map e_i. The features e_1 extracted from the reference image x_1 are then warped to i-th image to compute the residual $e_i - \phi_{m_i}(e_1)$, which indicates possible alignment errors. In parallel, the per-pixel noise estimate n_i is passed through a convolution+ReLU block, followed by a residual block to obtain 32-dimensional noise features $\tilde{n}_i$. Additionally, we pass the modulo 1 of scene motion m_i mod 1 through a convolution+ReLU block to obtain the 8-dimensional motion features $\tilde{m}_i$. The encoding e_i, the residual $e_i - \phi_{m_i}(e_1)$, the noise features $\tilde{n}_i$, and the motion features $\tilde{m}_i$ are then concatenated along the channel dimension and projected to 32 dimensional feature map using a convolution+ReLU block. The resulting feature map is passed through 1 residual blocks, followed by a final convolution layer to obtain the output $\tilde{v}_i$. Certainty value v_i for the i-th image is then obtained as the absolute value $|\tilde{v}_i|$ to ensure positive certainties.

Decoder D: The decoder passes the encoding $\hat{z}$ of the output image $\hat{y}$ through a convolution+ReLU block to obtain a 64-dimensional feature map. This feature map is passed through 9 residual blocks, followed by a final convolution layer which predicts the denoised image $\hat{y}$.

Motion Estimation in Ours†: In Section 5.2 in the main paper, we report results for a variant of our approach **Ours†** which employs a custom optical flow network to find the relative motion between image x_i and the reference image x_1. We use a pyramidal approach with cost volume, commonly employed in state-of-the-art optical flow networks [6, 3]. The architecture of our optical flow network is described here. We first pass both x_i and x_1 through a convolution+ReLU block to obtain 32-dimensional feature maps. These are then passed through 6 residual blocks. Before each of the first two residual blocks, we downsample the input feature maps by a factor of 2 using a convolution+ReLU block with stride 2 for computational efficiency. Next, we construct a feature pyramid with 2 scales, which is used to compute the optical flow. We pass the output of the last residual block through a convolution+ReLU block to obtain 64-dimensional feature maps f_i^1 and f_1^1. These feature maps are then passed through another convolution+ReLU block with stride 2 to obtain lower resolution feature maps f_i^2 and f_1^2. Next, we construct a partial cost volume containing pairwise matching scores between pixels in f_i^2 and f_1^2 using the correlation layer. For efficiency, we only compute matching scores of a pixel in f_i^2 with spatially nearby pixels in f_1^2 within a 7×7 window. The cost volume is concatenated with the feature map f_i^2 and passed through two convolution+ReLU blocks with 128 and 64 output dimensions. The output feature map is passed through a final convolution layer to obtain a coarse optical flow m_i^2. This initial estimate is upsampled by a factor of 2 and used to warp the feature map f_1^1 to the i-th image. We then compute the matching scores between f_i^1 and $\phi_{m_i^2}(f_1^1)$ using a 5×5 spatial window. A refined optical flow m_i^1 is then obtained using the same architecture as employed for pyramid level 2, without weight-sharing. The estimate m_i^1 is then upsampled by a factor of 4 to obtain the final motion estimate m_i.

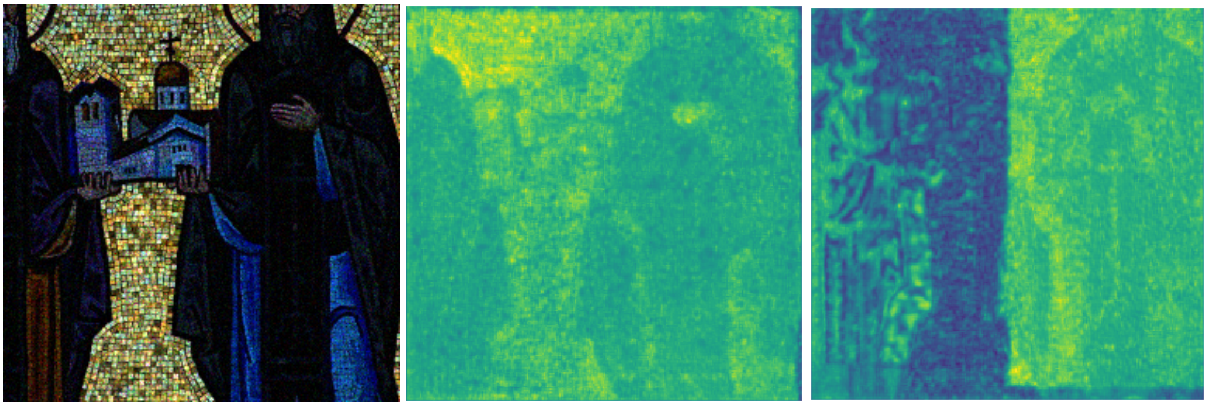
5. Analysis of Certainty Predictor

In this section, we analyse the behaviour of our certainty predictor W. The certainty predictor computes the certainty values v_i for each element in our residual $E(x_i) - G(\phi_{m_i}(z))$. This allows us to reduce the impact of *e.g.* errors in motion estimate m_i, by assigning a lower weight for such regions in our MAP objective (7) in the main paper. We analyse the behaviour of W by manually corrupting the motion estimate m_i. Figure 1b shows the channel-wise mean of the predicted certainty values v_i for an input image (Figure 1a), using the estimated scene motion m_i. We observe that the mean certainty values are approximately uniform over the image, with some slight variations according to the image intensity values. Next, we corrupt the motion estimate m_i for the left half of the image by adding a fixed offset of 16 pixels in both directions. The certainty values predicted using these corrupted motion m_i is shown in Figure 1c. As desired, our certainty predictor detects the alignment errors and assign a lower certainty values to the corresponding image regions (left half of the image).

6. Detailed Ablative Study

In this section, we provide a detailed ablative study analysing the impact different components in our architecture. Our analysis is performed on the SyntheticBurst super-resolution dataset [1], as well as the grayscale burst denoising dataset [4].

Impact of number of iterations K_{SD}: We analyse the impact of the number of steepest-descent iterations K_{SD} by training and evaluating our approach with different values of K_{SD}. The results on the SyntheticBurst super-resolution dataset [1] are provided in Table 1a, while the results on the grayscale burst denoising dataset [4] is shown in Table 1b. Additionally, a convergence analyses of the steepest-descent steps is provided in Figure 2. The entries with $K_{\text{SD}} = 0$ directly output the initial encoding z^0 predicted by our initializer P to the decoder. Since the initializer only utilizes the first burst image, the entry $K_{\text{SD}} = 0$ corresponds to the SingleImage baseline included in Table 1 and Table 2 in the main paper. Performing just a



(a) Input Image (b) Certainty Values, Original m_i (c) Certainty Values, Corrupted m_i

Figure 1: Analysis of our certainty predictor W. The channel-wise mean of the certainty values predicted by W for the input image (a), using the estimated scene motion m_i is shown in (b). Next, we corrupt the motion estimate m_i for the left half of the image by adding a fixed offset of 16 pixels in both directions. The mean certainty values predicted using the corrupted motion estimate m_i is shown in (c). Our certainty predictor can detect the errors in motion estimate m_i and assign lower certainty values to the corresponding image regions.

P	K_{SD}	PSNR↑	LPIPS↓	SSIM↑
	3	39.79	0.073	0.951
✓	0	36.29	0.123	0.912
✓	1	39.60	0.074	0.950
✓	2	39.75	0.074	0.951
✓	3	39.82	0.071	0.952
✓	4	39.77	0.071	0.951
✓	5	39.64	0.072	0.950

(a) SyntheticBurstSR

P	K_{SD}	Gain $\propto 1$	Gain $\propto 2$	Gain $\propto 4$	Gain $\propto 8$	Average
	3	39.29	36.42	33.33	28.86	34.47
✓	0	35.16	32.27	29.34	25.81	30.65
✓	1	39.32	36.48	33.38	29.39	34.64
✓	2	39.36	36.49	33.37	29.38	34.65
✓	3	39.37	36.51	33.38	29.69	34.74
✓	4	39.33	36.46	33.37	29.11	34.57
✓	5	39.42	36.54	33.40	28.25	34.40

(b) Grayscale Denoising

Table 1: Impact of initializer P and number of steepest-descent iterations K_{SD} on the SyntheticBurst super-resolution (a) and grayscale denoising (b) datasets.

single steepest-descent step already provides a large improvement over the SingleImage baseline with a PSNR of 39.60 on the SyntheticBurst dataset. This demonstrates the fast convergence of the steepest-descent iterations. Both the super-resolution and denoising performance improves gradually with an increase in K_{SD}, and the best results are obtained with $K_{\text{SD}} = 3$ iterations. Performing more iterations $K_{\text{SD}} > 3$ results in a small decrease in performance, indicating that early-stopping can act as a regularizer, leading to improved performance.

Impact of initializer P: Here, we analyse the impact of our initializer module P, which computes the initial encoding z^0. We evaluate a variant of our approach which does not utilizes P, instead setting the initial encoding z^0 to zeros. The results on the SyntheticBurst dataset and the grayscale burst denoising dataset are shown in Table 1a and Table 1b, respectively. Thanks to the fast convergence of the steepest-descent iterations, we observe that the initializer module P only provides a small improvement in performance.

Impact of downsampling factor $\tilde{s}$ of feature degradation G: We analyse the impact of the downsampling factor $\tilde{s}$ of operator G on the burst super-resolution task. Results for different values of $\tilde{s}$ on the SyntheticBurst dataset is provided in Table 2. We observe that using a higher downsampling factor $\tilde{s}$ in operator G leads to better results. However the improvement when using a downsampling factor $\tilde{s} = 4$ compared to $\tilde{s} = 2$ is relatively small (+0.07dB). Hence, we use $\tilde{s} = 2$ in our final version to obtain better computational efficiency.

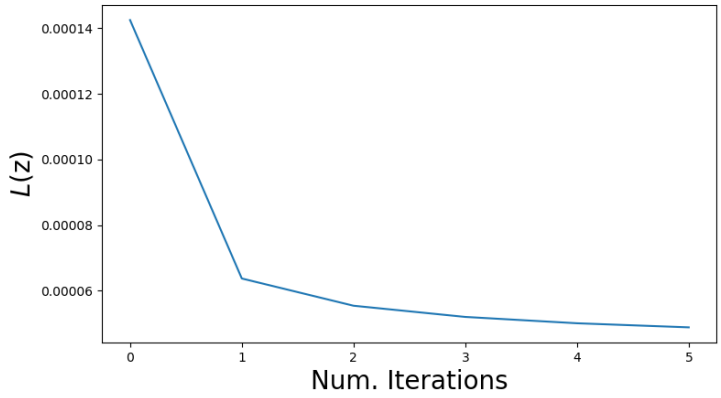


Figure 2: Convergence analysis of the steepest-descent iterations. We plot our loss (7) in the main paper, w.r.t. the number of steepest-descent iterations. The loss is averaged over the 300 burst sequences from the SyntheticBurst dataset.

$\tilde{s}$	PSNR↑	LPIPS↓	SSIM↑
1	39.38	0.077	0.948
2	39.82	0.071	0.952
4	39.89	0.071	0.953

Table 2: Impact of downsampling factor $\tilde{s}$ of feature degradation G on the SyntheticBurst dataset.

7. Qualitative Results

In this section, we provide additional qualitative results. A qualitative comparison with BPN [7] on the grayscale [4] and color [7] burst denoising datasets are provided in Figure 3 and Figure 4, respectively. Figure 5 contains a comparison of our approach to DBSR [1] on the SyntheticBurst RAW super-resolution dataset. Additional comparison on the real-world BurstSR dataset [1] is provided in Figure 6.

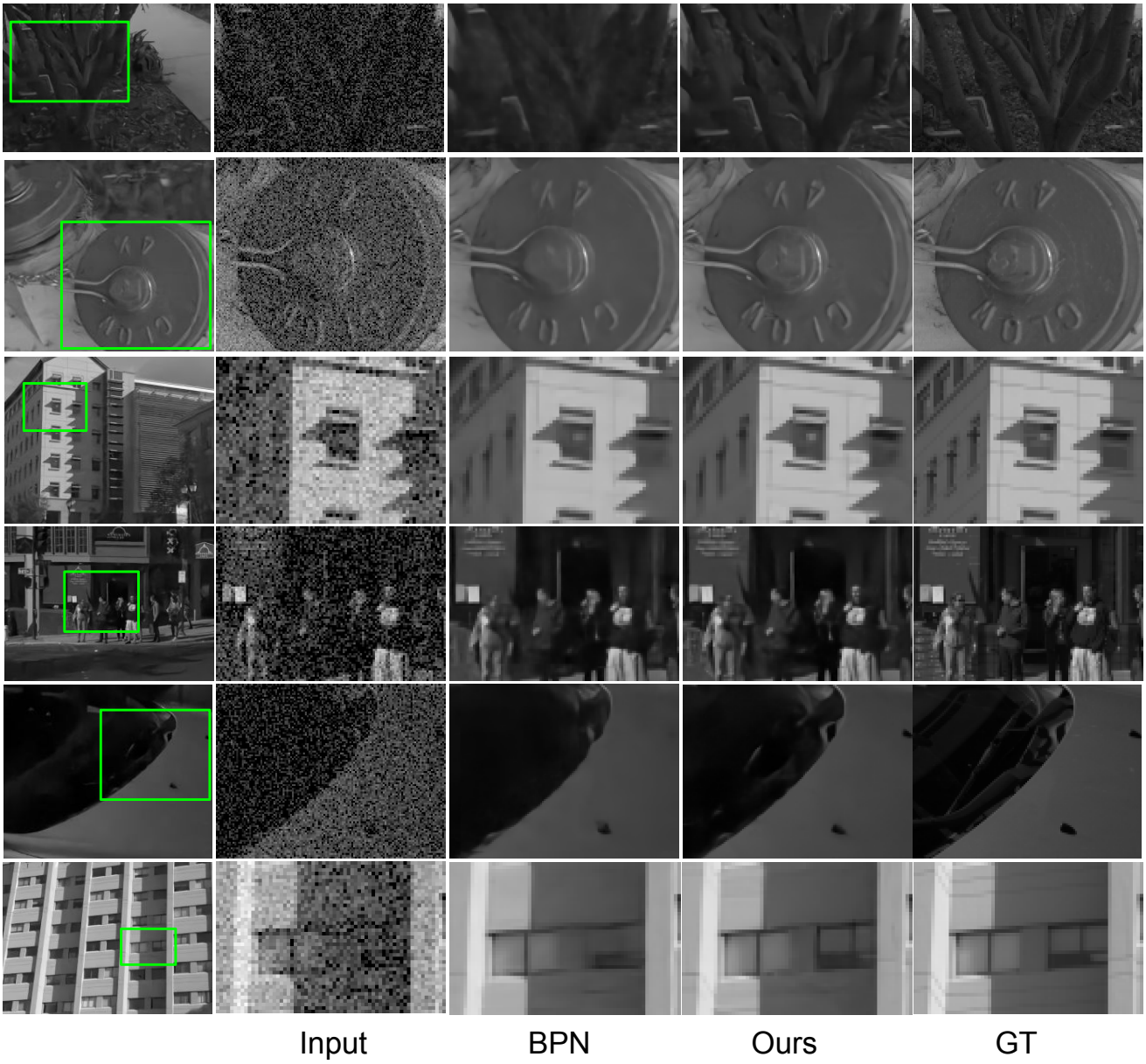


Figure 3: Qualitative comparison of our approach with BPN [7] on the grayscale burst denoising dataset [4]. Compared to BPN, our approach can recover higher frequency details, without oversmoothing the image.

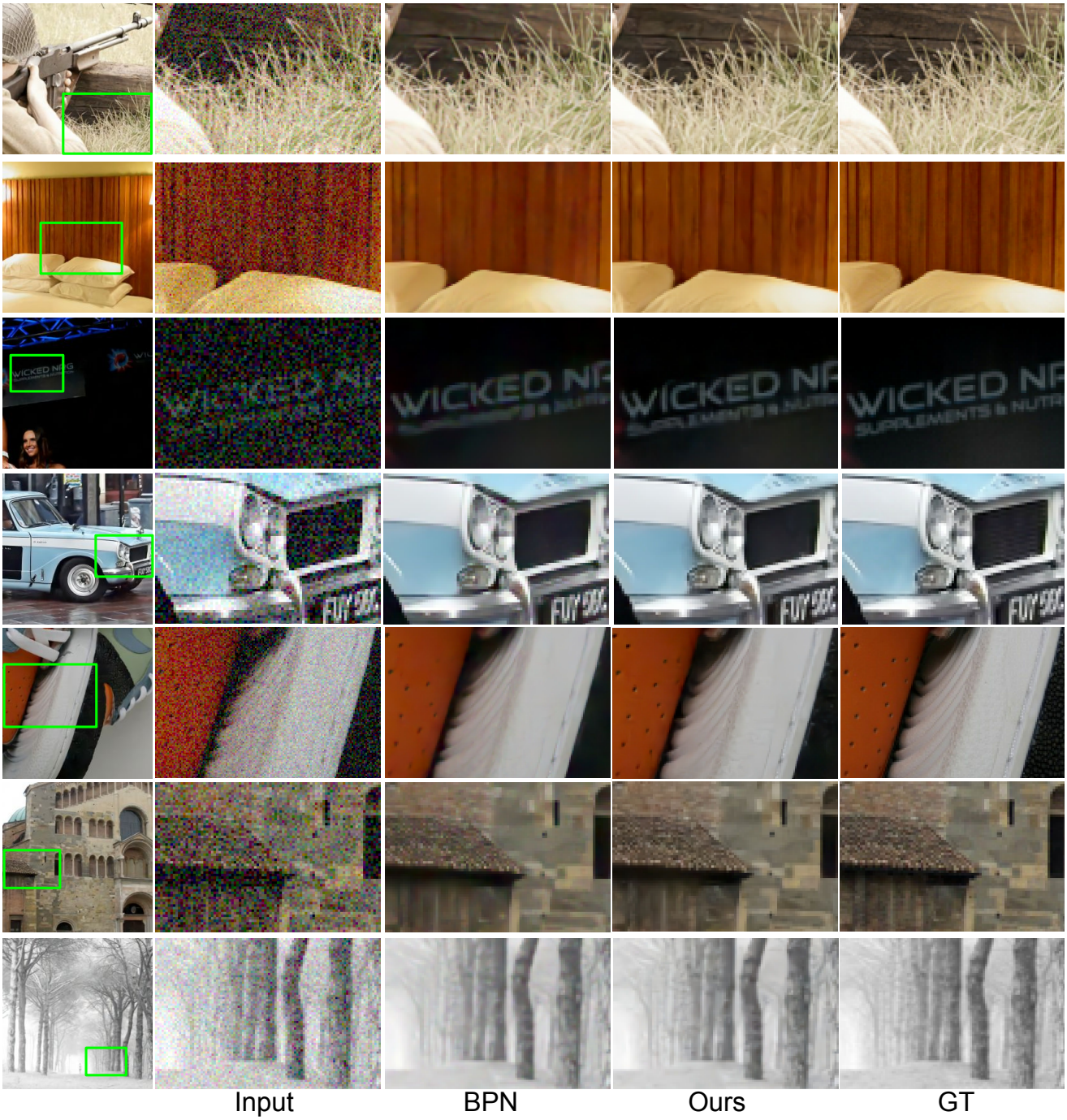


Figure 4: Qualitative comparison of our approach with BPN [7] on the color burst denoising dataset [7]. Our approach can generate clean images without introducing any color artifacts.

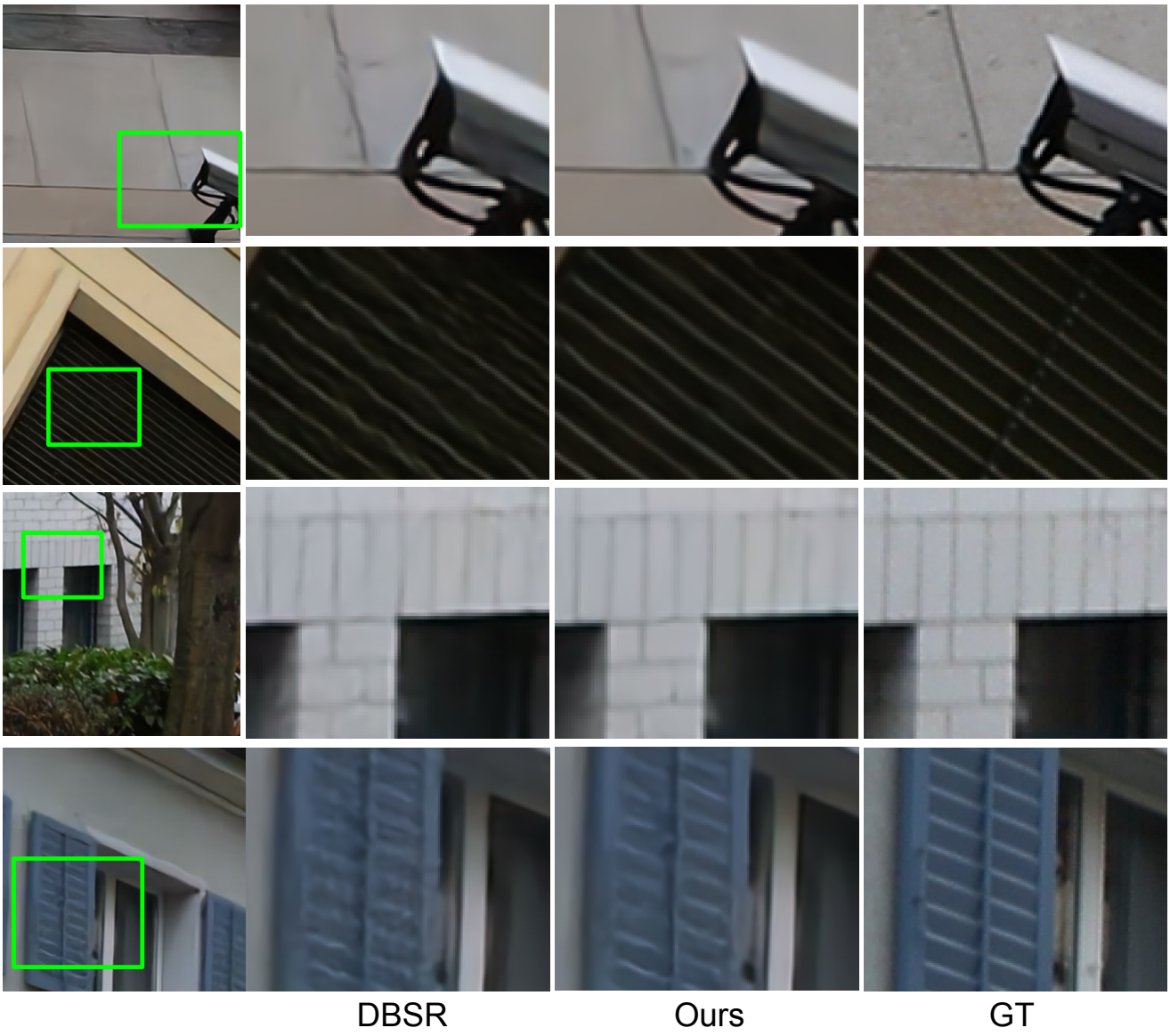


Figure 5: Qualitative comparison of our approach with DBSR [1] on the SyntheticBurst super-resolution dataset [1]. Our approach can better recover high frequency details and generates sharper images.

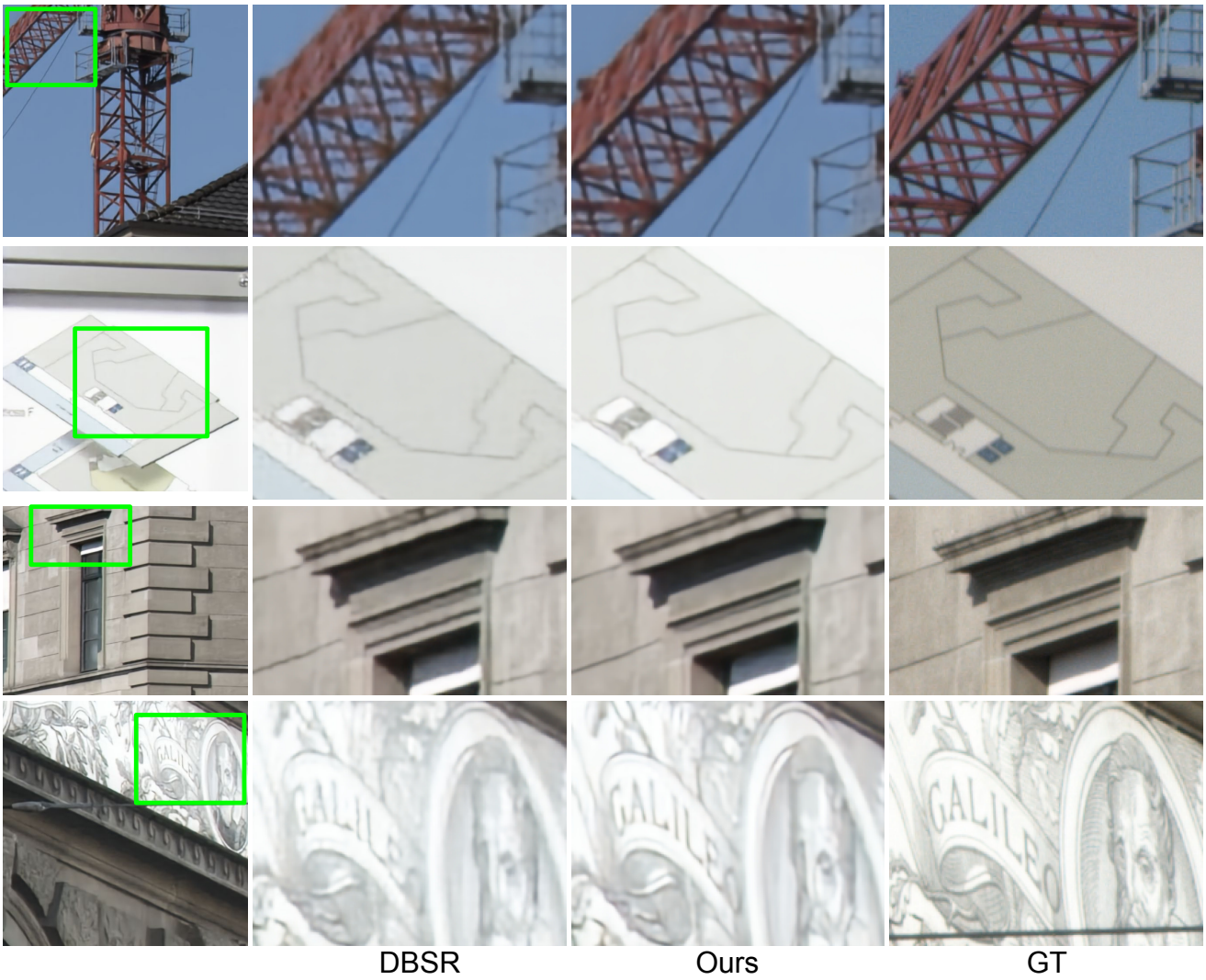


Figure 6: Qualitative comparison of our approach with DBSR [1] on the real-workd BurstSR dataset [1]. Note that the ground truth image and the input burst are captured using different cameras, resulting in a color shift between the network predictions and the ground truth.

References

[1] Goutam Bhat, Martin Danelljan, L. Gool, and R. Timofte. Deep burst super-resolution. In *CVPR*, 2021. 4, 6, 9, 10

[2] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 3

[3] Eddy Ilg, N. Mayer, Tonmoy Saikia, Margret Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1647–1655, 2017. 4

[4] Ben Mildenhall, J. Barron, Jiawen Chen, Dillon Sharlet, R. Ng, and Robert Carroll. Burst denoising with kernel prediction networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2502–2510, 2018. 4, 6, 7

[5] W. Shi, J. Caballero, Ferenc Huszár, J. Totz, A. Aitken, R. Bishop, D. Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016. 3

[6] Deqing Sun, X. Yang, Ming-Yu Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018. 4

[7] Zhihao Xia, Federico Perazzi, M. Gharbi, Kalyan Sunkavalli, and A. Chakrabarti. Basis prediction networks for effective burst denoising with large kernels. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11841–11850, 2020. 6, 7, 8