
DeblurDiff: Real-World Image Deblurring with Generative Diffusion Models

- Supplemental Material -

Lingshun Kong^{1,*}, Jiawei Zhang², Dongqing Zou³, Fu Lee Wang⁴,
Jimmy S. Ren⁴, Xiaohe Wu⁵, Jiangxin Dong¹, Jinshan Pan^{1,†}

¹Nanjing University of Science and Technology ²SenseTime Research ³PBVR

⁴Hong Kong Metropolitan University ⁵Harbin Institute of Technology

1 Details of the training dataset

As mentioned in the main paper, existing deblurring datasets such as GoPro [6] and DVD [10] have several limitations. These datasets typically contain a relatively small number of images (usually only a few thousand) and a limited variety of scenes (often only a few dozen). This insufficiency makes it difficult to train a robust deblurring diffusion model. Moreover, for the GoPro dataset, which uses multi-frame video sequences to synthesize blur, the low frame rate of the videos results in blur patterns that significantly differ from real-world blur. Therefore, we do not train our model on these problematic datasets. Instead, we constructed a large-scale dataset to train our DeblurDiff model. This dataset addresses the limitations of existing datasets by providing a richer and more diverse set of image pairs, thereby better supporting the learning and generalization capabilities of the model.

Our training dataset consists of three parts: (1) Existing deblurring datasets, including MC-Blur [15] and RSBlur [9], which contain high-resolution blur-sharp data pairs (approximately 100,000 images). Some of their training data pairs are illustrated in Figure 1. (2) We capture some high-frame-rate high-definition video clips, generating blurred and clear data pairs (approximately 200,000 images) using the same strategy as REDS [7]. We present some of our clear and blurred data pairs in Figure 2. (3) We collected a large number of high-definition images as ground truth (approximately 200,000 images) and generated various motion blur kernels to synthesize corresponding blurred images. Specifically, using the method proposed by [1], we generated 100,000 motion blur kernels (some of which are presented in Figure 3). The size of the motion kernel is 63×63 , and we randomly generate motion trajectories within this range, with the length of the trajectory being a random number between 1 and 1000. We use these kernels and high-definition images to synthesize two types of blurred and clear data pairs: globally uniform blur and non-uniform blur. For globally uniform blur, we directly use the generated kernel as the blur kernel to convolve with the clear image, obtaining the corresponding blurred image. For non-uniform blur, we start with two different sharp images. We use the SAM2 method [8] to segment objects in one of the sharp images, identifying the foreground objects. The background is then taken from the second sharp image. We apply different motion blur kernels to the segmented foreground objects and the background separately. Finally, we composite the blurred foreground objects onto the blurred background image to create the final non-uniformly blurred image, simulating the inconsistent blur caused by the motion of different objects in the real world. We present the training data pairs for globally uniform blur and non-uniform blur in Figures 4 and Figure 5, respectively.

*This project is done during the internship at SenseTime Research.

†Corresponding author



Figure 1: The blurred and clear training data pairs in the RSBlur dataset [9].



Figure 2: The blurred and clear training data pairs synthesized using consecutive video frames.

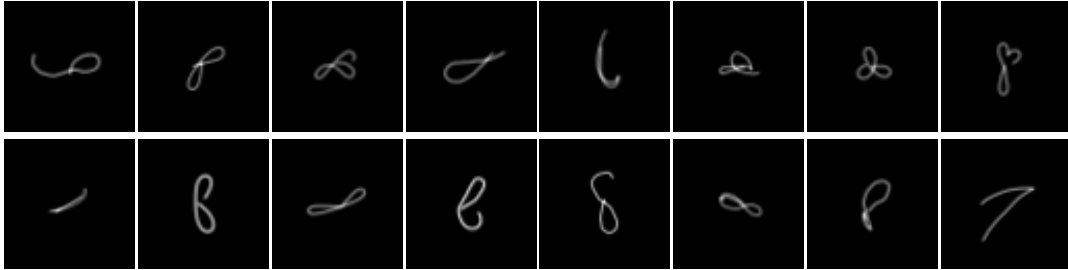


Figure 3: Motion blur kernels generated by [1].



Figure 4: The data pairs of globally uniform blurred images and clear images.



Figure 5: The data pairs of globally non-uniform blurred images and clear images.

2 Comparison with More Methods.

For a fair comparison, we include additional comparisons with more single-image methods (including Restormer [13], NAFNet [2]) in Table 1. where "*" indicates that the method is retrained on our dataset. NAFNet and FFTformer represent the state-of-the-art among CNN-based methods and Transformer-based methods. Our method consistently outperforms both in most no-reference metrics across various datasets.

Table 1: Quantitative evaluations of the proposed method against state-of-the-art ones on both synthetic and real-world benchmarks. The models marked with an asterisk * indicate that we retrain them on our own training set. The best and second performances are marked in **red** and **blue**, respectively. For the RWBI and Real Blurry Images datasets, which lack ground truth (GT) data, we evaluate the performance using only no-reference metrics.

Dataset	Metrics	FFTformer	FFTformer*	DBGAN	Restormer	Restormer*	NAFNet	NAFNet*	ResShift	ResShift*	HI-Diff	HI-Diff*	ControlNet*	PASD*	DiffBIR*	Ours
GoPro	PSNR ↑	34.21	26.86	31.18	32.92	23.77	33.71	25.34	29.03	24.86	33.33	25.90	22.31	22.82	23.86	24.32
	SSIM ↑	0.9536	0.8357	0.9182	0.9416	0.7466	0.9490	0.8144	0.8781	0.7838	0.9462	0.8097	0.6547	0.6559	0.7173	0.7375
	LPIPS ↓	0.0725	0.1538	0.1120	0.0863	0.2800	0.0805	0.1719	0.0780	0.2551	0.0820	0.2163	0.3292	0.2984	0.2772	0.2191
	FID ↓	6.7911	13.8896	10.7629	8.8983	36.0093	7.0050	17.8093	8.8820	38.6699	8.1553	25.2230	37.2749	39.0057	27.0576	17.6948
	KID ↓	0.0006	0.0009	0.0010	0.0009	0.0065	0.0006	0.0009	0.0001	0.0066	0.0008	0.0026	0.0057	0.0054	0.0042	0.0009
	NIQE ↓	5.0338	4.1200	5.1988	5.1818	5.6205	5.0984	4.8230	4.8367	5.0208	4.6119	5.3965	3.5305	2.6567	3.4193	3.1769
	MUSIQ ↑	46.0739	52.2993	42.0985	44.9760	30.5318	45.2813	42.6742	44.2820	37.7094	47.7791	35.2030	59.4246	61.5345	56.3249	61.6369
	MANIQA ↑	0.5442	0.5454	0.4976	0.5304	0.3821	0.5394	0.4847	0.9419	0.4291	0.6119	0.4307	0.5746	0.5904	0.5464	0.6134
	CLIP-IQA ↑	0.4139	0.4360	0.3788	0.4034	0.3177	0.4115	0.3851	0.4229	0.3388	0.4841	0.3553	0.5869	0.5758	0.5260	0.5966
DVD	PSNR ↑	27.29	27.07	27.78	29.63	25.94	29.61	26.94	27.78	25.93	30.31	27.99	22.03	22.23	23.49	23.74
	SSIM ↑	0.8426	0.8534	0.8356	0.8806	0.8014	0.8850	0.8406	0.8420	0.8012	0.8972	0.8525	0.6409	0.6440	0.7114	0.7055
	LPIPS ↓	0.1993	0.1628	0.2126	0.1623	0.2056	0.1618	0.1587	0.1249	0.2312	0.1363	0.1725	0.3330	0.2968	0.2795	0.2501
	FID ↓	13.4680	6.7968	14.9420	8.1618	12.3697	7.7119	7.9729	7.1787	18.6424	5.7370	9.3141	24.1213	22.3637	20.2467	12.5545
	KID ↓	0.0018	0.0002	0.0022	0.0005	0.0006	0.0007	0.0008	0.0002	0.0036	0.0001	0.0002	0.0029	0.0032	0.0032	0.0009
	NIQE ↓	4.7344	3.8562	4.8188	4.7266	4.5231	4.7265	4.0889	4.6422	4.4862	5.1858	4.4312	3.4037	3.2504	3.1357	2.7822
	MUSIQ ↑	40.0924	60.1091	40.4781	44.1288	45.1593	44.3610	51.8058	52.7551	57.6779	45.5395	49.3134	65.3657	68.3299	61.6415	67.2447
	MANIQA ↑	0.5653	0.6257	0.5548	0.5920	0.5655	0.5919	0.5983	0.5744	0.6083	0.5360	0.5904	0.6155	0.6409	0.5773	0.6480
	CLIP-IQA ↑	0.4454	0.5271	0.4346	0.4648	0.4532	0.4627	0.4760	0.4831	0.5143	0.4065	0.4764	0.6606	0.6528	0.5829	0.6686
Realblur	PSNR ↑	29.97	26.94	23.91	26.63	27.31	26.33	26.99	26.30	29.67	30.18	27.47	23.77	25.02	25.50	25.71
	SSIM ↑	0.9036	0.8580	0.7434	0.8428	0.8611	0.8261	0.8616	0.8140	0.8857	0.9049	0.8711	0.6787	0.7642	0.7724	0.7705
	LPIPS ↓	0.0906	0.1411	0.2945	0.1549	0.1622	0.1675	0.1481	0.1249	0.1502	0.0868	0.1332	0.2565	0.2075	0.1951	0.1693
	FID ↓	11.7184	19.8415	99.5935	24.0475	24.8902	29.1702	21.3381	21.6440	38.1609	11.4180	18.9316	40.4691	38.7507	30.4657	22.7713
	KID ↓	0.0008	0.0011	0.0212	0.0015	0.0016	0.0021	0.0013	0.0006	0.0035	0.0007	0.0008	0.0025	0.0026	0.0025	0.0007
	NIQE ↓	5.1833	4.3473	5.3228	5.2270	5.2486	5.0476	5.0345	5.2628	5.1109	5.1437	4.9070	4.6525	3.9192	4.3053	4.2666
	MUSIQ ↑	52.2186	61.5808	38.4866	48.4321	51.3666	47.0689	57.7496	49.3209	51.8276	57.1640	58.4964	66.5174	61.1498	58.7450	65.0557
	MANIQA ↑	0.6165	0.6374	0.4322	0.5597	0.5543	0.5488	0.5998	0.5373	0.5584	0.6218	0.6114	0.6452	0.5994	0.5869	0.6538
	CLIP-IQA ↑	0.5077	0.5336	0.3469	0.4388	0.4534	0.4236	0.4682	0.4521	0.4984	0.5101	0.5258	0.6041	0.5457	0.5261	0.6087
RWBI	NIQE ↓	4.9704	4.4631	5.2905	5.2947	6.1161	4.9706	4.8877	5.4446	6.4613	5.3373	6.0858	5.0331	4.1973	4.2742	4.5171
	MUSIQ ↑	43.0404	59.6223	42.7631	44.4358	39.1635	46.6243	51.3378	51.0359	52.3279	47.1820	47.2667	62.5079	62.1680	61.8865	66.7505
	MANIQA ↑	0.5201	0.5425	0.4852	0.5045	0.4391	0.4885	0.5072	0.4953	0.5297	0.5082	0.4835	0.5758	0.5645	0.5618	0.6260
	CLIP-IQA ↑	0.3749	0.5413	0.3645	0.3860	0.3560	0.3800	0.4342	0.5032	0.5199	0.3907	0.3861	0.6199	0.5820	0.6042	0.6849
Real Images	NIQE ↓	4.4877	3.8520	4.9338	5.1274	5.1951	4.2941	4.4452	5.4704	5.2451	4.7018	4.5129	4.0978	4.4460	3.7964	3.6628
	MUSIQ ↑	38.1174	52.9290	32.0568	31.4229	41.6138	39.1193	48.4242	48.8154	50.8100	43.8702	45.1412	51.5191	61.6320	53.6088	52.9263
	MANIQA ↑	0.5487	0.5170	0.4488	0.4966	0.5357	0.4942	0.5615	0.5345	0.5618	0.5722	0.5573	0.5544	0.5937	0.5564	0.5963
	CLIP-IQA ↑	0.4222	0.5026	0.3501	0.3994	0.4287	0.3914	0.4531	0.4767	0.4974	0.4545	0.4617	0.5254	0.5919	0.5384	0.5496

3 The specific architecture of the DeblurDiff.

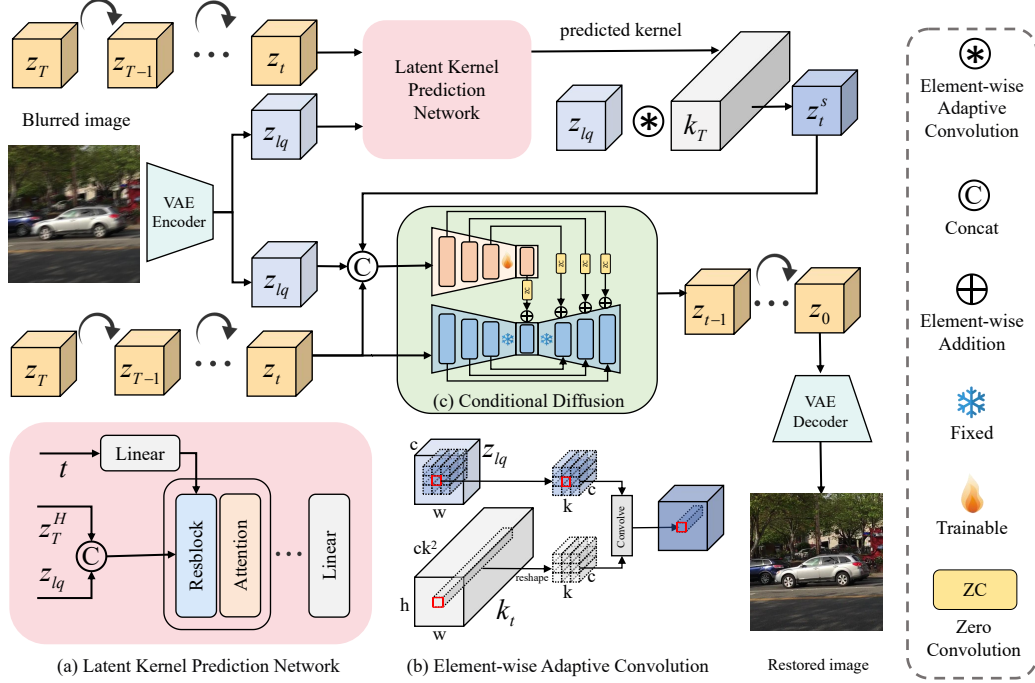


Figure 6: Overall architecture of the proposed method.

LKPN is a UNet structure with 4 layers of encoders and decoders, each layer comprising two blocks, which consist of a ResBlock [4] and a self-attention [11] layer. Figure 6(a) shows the specific structures of LKPN. The input to the LKPN is formed by concatenating z_t and z_{lq} . Additionally, the time step t is embedded into the same dimension using a linear layer and added before the out_layer of the ResBlock. Given an input z_{lq} with a shape of (c, h, w) after encoding with the VAE encoder. After passing through the LKPN's UNet, the shape of the output features remains (c, h, w) . At the end of the LKPN, a linear layer is used to transform the shape into $(c \times k \times k, h, w)$, where k represents the size of the spatially variant kernel we estimate. In this paper, we use $k = 5$.

EAC is used to restore the latent of the clear image while preserving the input information by utilizing the spatially-variant kernel estimated by the LKPN. Figure 6(b) shows the specific structures of EAC.

The EAC first reshapes the kernel estimated by the LKPN into (h, w, ck^2) . For each position (h_i, w_i, c_i) of $z_{lq} \in \mathbb{R}^{h \times w}$, a spatially kernel $\mathcal{F}_{h_i, w_i, c_i} \in \mathbb{R}^{k \times k}$ is applied to the region centered around $z_{lq}(h_i, w_i, c_i)$ as follows:

$$\begin{aligned} \hat{z}_{lq}(h_i, w_i, c_i) &= \mathcal{F}_{h_i, w_i, c_i} * z_{lq}(h_i, w_i, c_i) \\ &= \sum_{n=-r}^r \sum_{m=-r}^r \mathcal{F}(h_i, w_i, k^2 c_i + kn + m) \\ &\quad \times z_{lq}(h_i - n, w_i - m, c_i), \end{aligned} \quad (1)$$

where $r = \frac{k-1}{2}$, $*$ denotes convolution operation, \mathcal{F} is the generated filter, $z_{lq}(h_i - n, w_i - m, c_i)$ and $\hat{z}_{lq}(h_i, w_i, c_i)$ denote the input features and transformed features, respectively.

Conditional Diffusion is used to enhance the capabilities of pre-trained SD models by introducing additional conditional inputs to precisely control the image generation process. Figure 6(c) shows the specific structures of Conditional Diffusion. Specifically, the method guides the diffusion process by integrating a conditioning input (e.g., a blurry image) with the noisy latent representation from the diffusion model. Initialized with pre-trained diffusion model weights, an auxiliary network employs ZeroConv layers (1x1 convolutions initialized with zero weights) to gradually incorporate the conditioning signal into the diffusion process, without disrupting the pre-trained knowledge. This allows the model to iteratively refine the latent representation, generating high-quality outputs

that align with the provided guidance. We use the concatenated z_{lq} and z^s obtained from EAC as conditional inputs. For the first layer, we do not use the weights from the original SD model for initialization, since our input channels differ from those of the original SD. Instead, we used random initialization.

The sampling process of DeblurDiff is shown in Algorithm 1.

Algorithm 1 DeblurDiff Sampling

Input: Blurry image X_B
Output: Deblurred image X_D
 $\mathbf{z}_{lq} = \mathcal{E}(\mathbf{X}_B)$
 $\mathbf{z}_T \sim \mathcal{N}(0, I)$
for $t = T, \dots, 1$ **do**
 $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
 $k_t = \text{LKPN}(\mathbf{z}_t, \mathbf{z}_{lq}, t)$
 $z_t^s = \text{EAC}(\mathbf{z}_{lq}, k_t)$
 $\epsilon_{pred,t} = \text{ControlNet}(z_t^s, z_{lq}, z_t, t)$
 $\mathbf{z}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{z}_t - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}} \epsilon_{pred,t} \right) + \sigma_t \mathbf{z}$
end for

4 More experimental results

Poor quality of the GoPro dataset. For the existing evaluation dataset, such as GoPro [6], the quality of the ground truth (GT) itself is not particularly high. As a result, metrics like PSNR and SSIM have limited significance in reflecting visual fidelity. We follow prior works [5, 12] and use the same evaluation metrics for comparison. Our method focuses on generating visually pleasing images with rich details, often producing results that are visually better than the GT, as shown in Figure 7. This preference for visual quality over strict pixel-wise accuracy leads to relatively lower PSNR/SSIM scores. In Table 2, we present a detailed comparison between our method and the GT, where no-reference metrics further indicate that our results exhibit better visual quality.

Table 2: The quantitative results of our method compared to the GT from GoPro dataset [6].

Result	NIQE	MUSIQ	MANIQA	CLIP-IQA
GT	4.0254	47.9752	0.5668	0.4308
Ours	3.6628	52.9263	0.5963	0.5496

Impact of the GoPro training dataset on deblurring results. We do not include the GoPro dataset in the training dataset when training the proposed method, as the quality of the GoPro dataset is not high. When these GTs with poor quality are included in the training dataset, it degrades the performance of the model’s generated results. We present in Table 3 the impact of including or excluding the GoPro training data on the deblurring results. When the training data does not include the GoPro dataset, the model achieves better no-reference metrics. Figure 8 shows that when the GoPro dataset is included in the training set, the generated results exhibit significant artifacts. Therefore, our method does not utilize the GoPro training set during training. Table 3: Quantitative evaluations of the impact of the GoPro dataset on deblurring results on the real blurry image [3].

Dataset	NIQE	MUSIQ	MANIQA	CLIP-IQA
Training dataset w/ GoPro	3.7162	51.9673	0.5642	0.5267
Training dataset w/o GoPro	3.6628	52.9263	0.5963	0.5496

Inference time. We provide runtime comparisons in Table 4. The size of the test image is 512×512 pixels. The test environment is based on a machine with an NVIDIA 4090D. Our method, DiffBIR, and ControlNet all employ 50 steps during inference. FFTformer is not a diffusion-based method, so it has a significant advantage in terms of running time. However, its visual quality is inferior to that of diffusion-based methods as shown in Figure 13- 22. The runtime of our method is slightly slower than that of ControlNet but faster than that of DiffBIR. The GPU memory



Figure 7: Visual comparison of our method’s results with the Ground Truth from GoPro. Our results have better visual effects than the Ground Truth.

usage of our method is slightly higher than that of ControlNet but lower than that of DiffBIR.

Table 4: Memory and running time comparisons of the compared methods.

Methods	FFTFormer	ControlNet	DiffBIR	Ours
Running time (s)	0.8	40	55	46
GPU memory (G)	6.1	7.6	8.0	7.8

Effectiveness of the deblur kernel generated by LKPN. An example of the deblur kernel (one channel selected for visualization purposes) is shown in Figure 9. The blur of the cyclist is different from that of the background, so the corresponding values of their kernels are also different. The kernels are then applied through the EAC, which adaptively addresses distinct blur characteristics at each pixel location, effectively preserving the input information and recovering sharp structures and fine details.

Selection of kernel size for LKPN. Table 5 shows the effect of kernel size k . The results demonstrate that while increasing k beyond 5 (to $k = 7$) yields no statistically significant performance gains, both $k = 5$ and $k = 7$ outperform $k = 3$. However, larger k values increase model complexity without com-

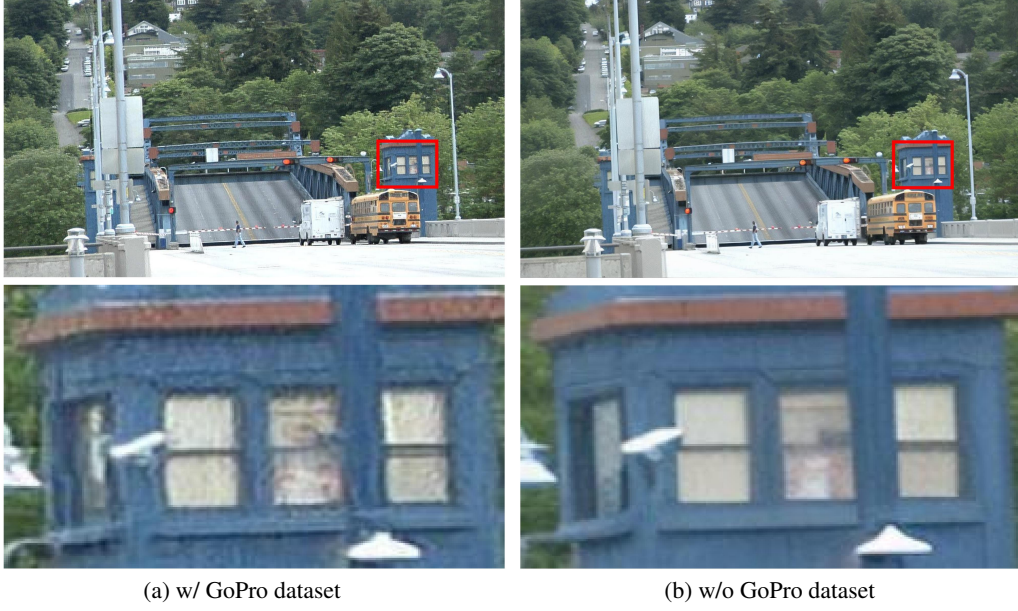


Figure 8: Visual comparison of our method’s results on different training datasets. When the training dataset includes GoPro training data, it degrades the visual quality of the results.

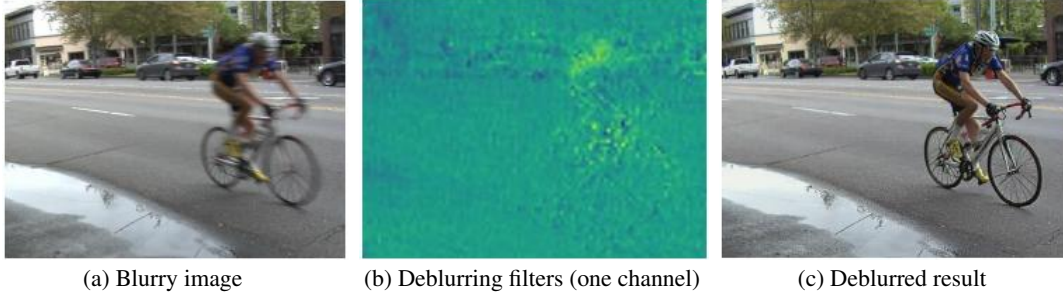


Figure 9: Visualization of the deblur kernel generated by LKPN.

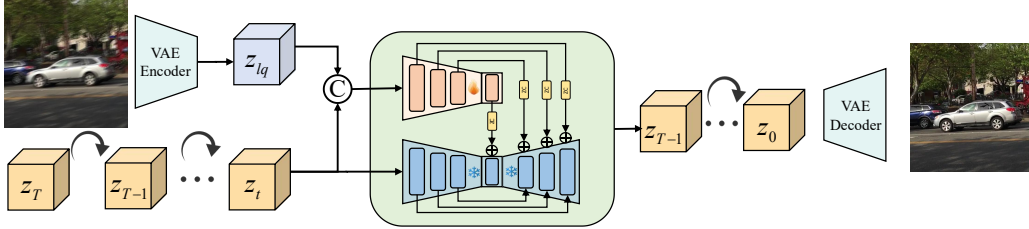
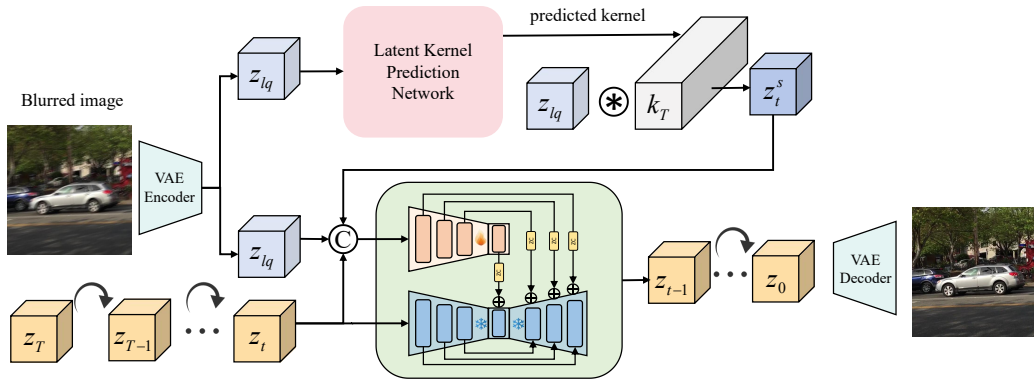
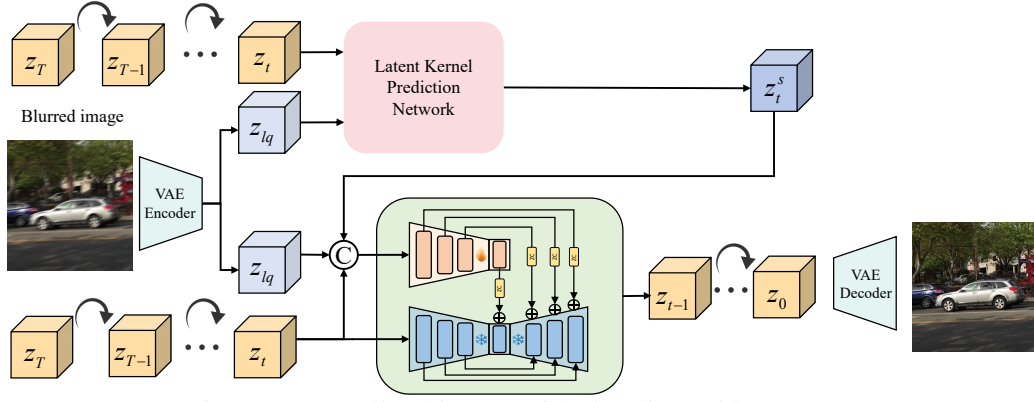


Figure 10: Overall architecture of ControlNet.

mensurate benefits. Based on this trade-off analysis, we selected $k = 5$ as the optimal configuration.

Table 5: Quantitative evaluations of the kernel size in LKPN on the real blurry image [3].

kernel size	NIQE	MUSIQ	MANIQA	CLIP-IQA
$k = 3$	3.8272	50.8526	0.5334	0.5173
$k = 5$	3.6628	52.9263	0.5963	0.5496
$k = 7$	3.6732	53.0145	0.5854	0.5517



5 More Visual Comparisons.

In this section, we provide more visual comparisons of the proposed method with state-of-the-art ones on both synthetic and real-world benchmarks in Figures 13 to 22.

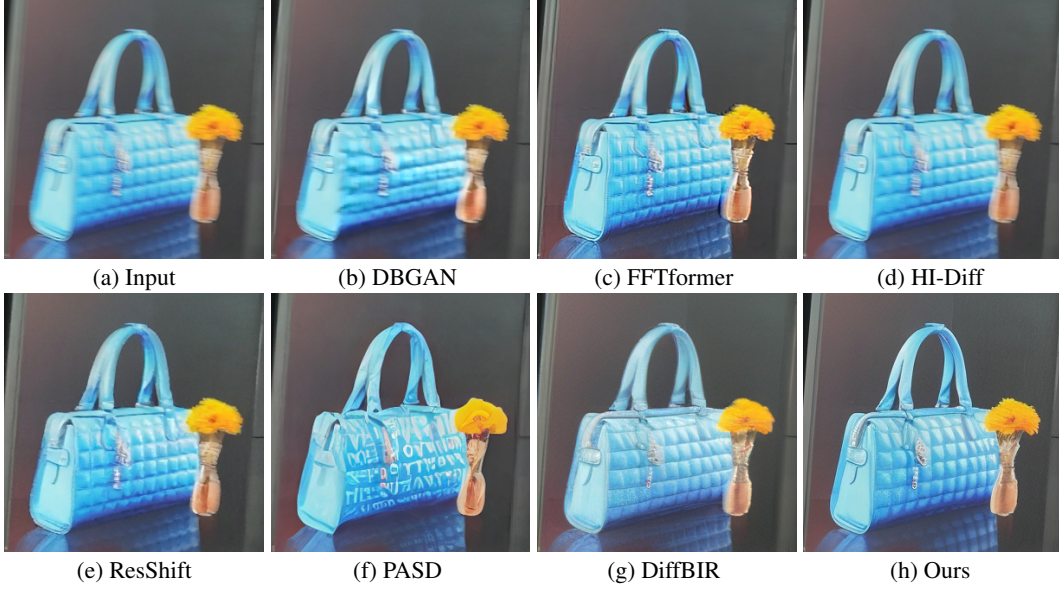


Figure 13: Deblurred results on the RWBI dataset [14]. The results in (b) to (g) fail to preserve the input information well while deblurring. In contrast, our method generates clear results while effectively retaining the input information.

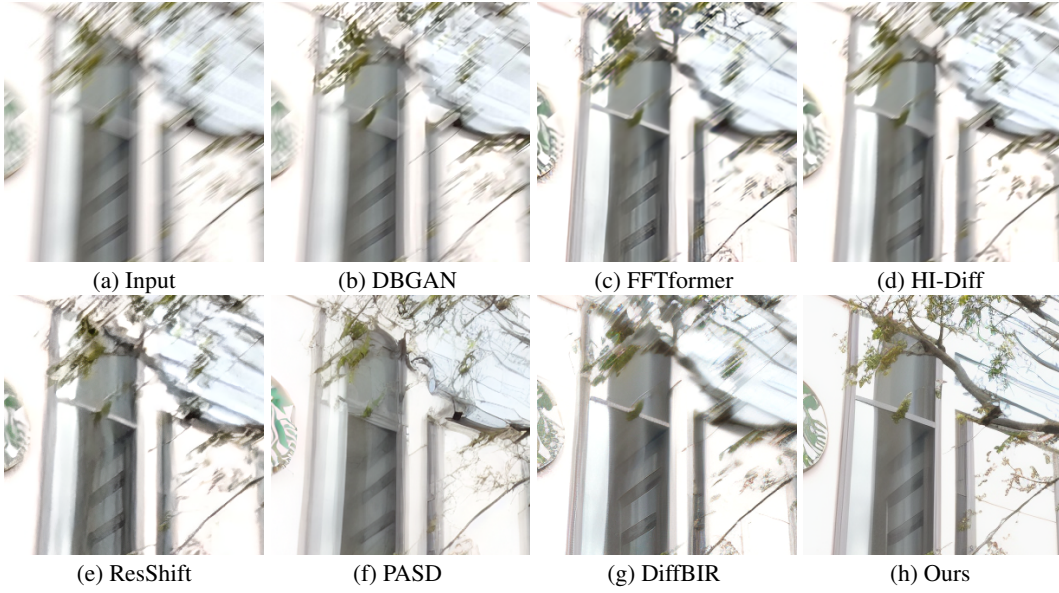


Figure 14: Deblurred results on the RWBI dataset [14]. The deblurred results in (b)-(g) still contain significant blur effects. The proposed method generates a clearer image, where the structure of the branches and the leaves are much clearer.

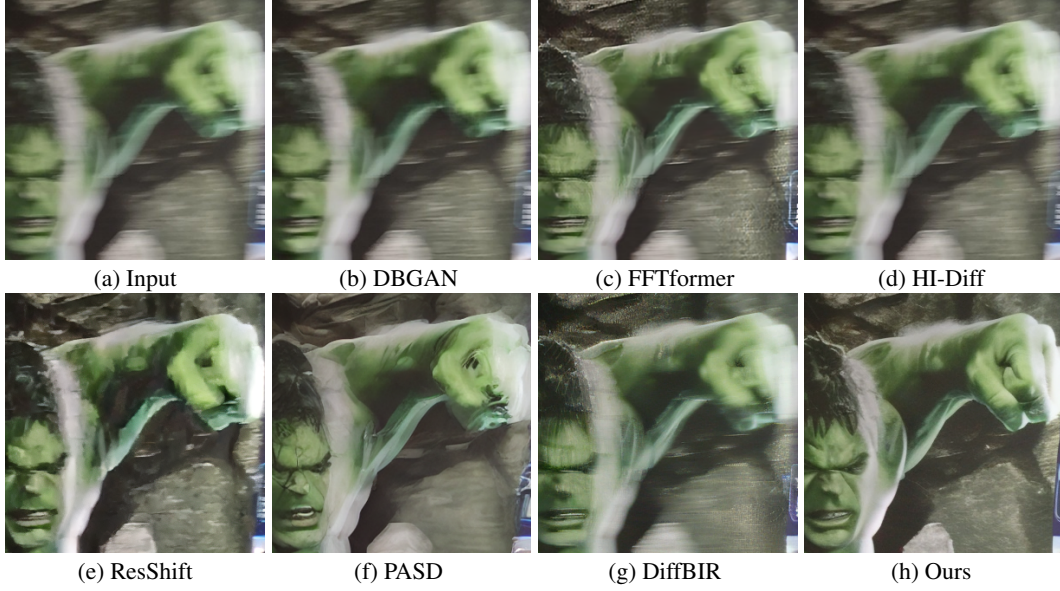


Figure 15: Deblurred results on the RWBI dataset [14]. The results in (b) to (g) leave severe artifacts. In contrast, our method generates a clear and realistic result.

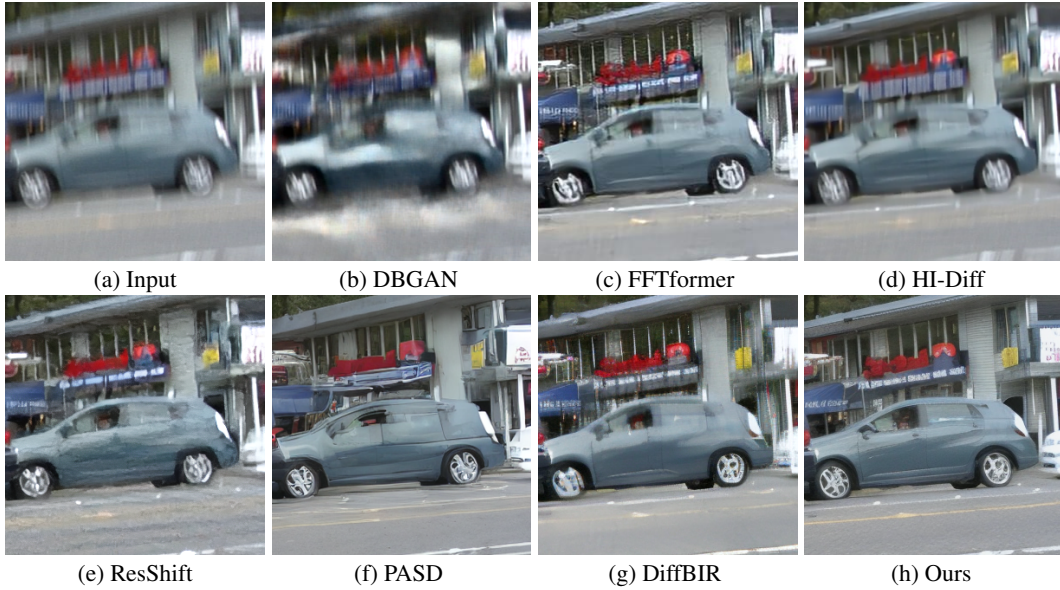


Figure 16: Deblurred results on the DVD dataset [10]. The structures in the results restored by methods (b) to (g) are all distorted. In contrast, the structures in our results are clear and accurate.



Figure 17: Deblurred results on the GoPro dataset [6]. The results in (b) to (d) fail to effectively restore the clear structure and details of the leaves, whereas our method recovers better details.

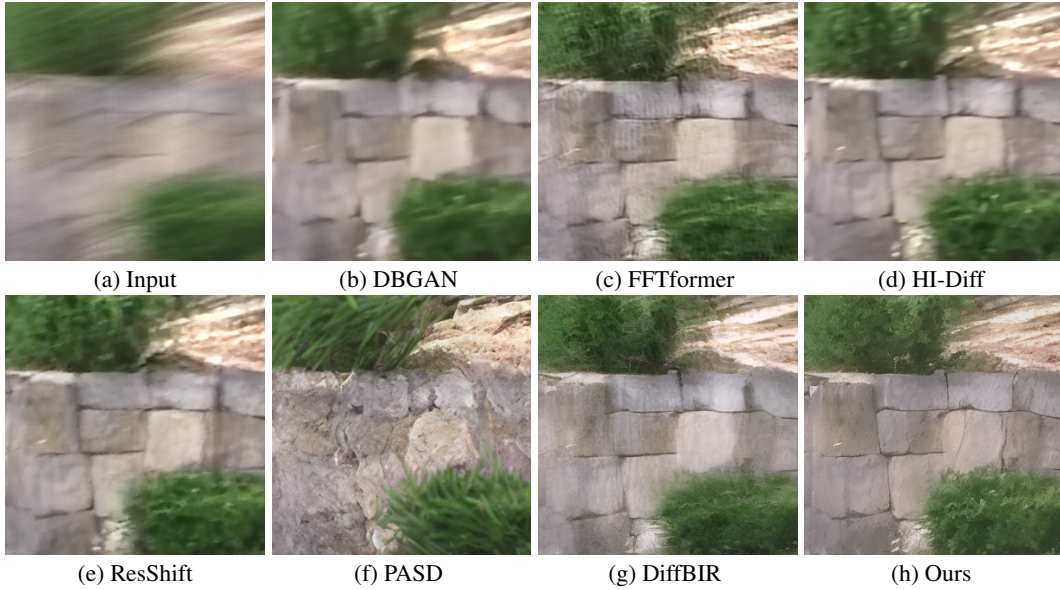


Figure 18: Deblurred results on the GoPro dataset [6]. The deblurred results in (b)-(g) still contain significant blur effects. The proposed method generates a clearer image.

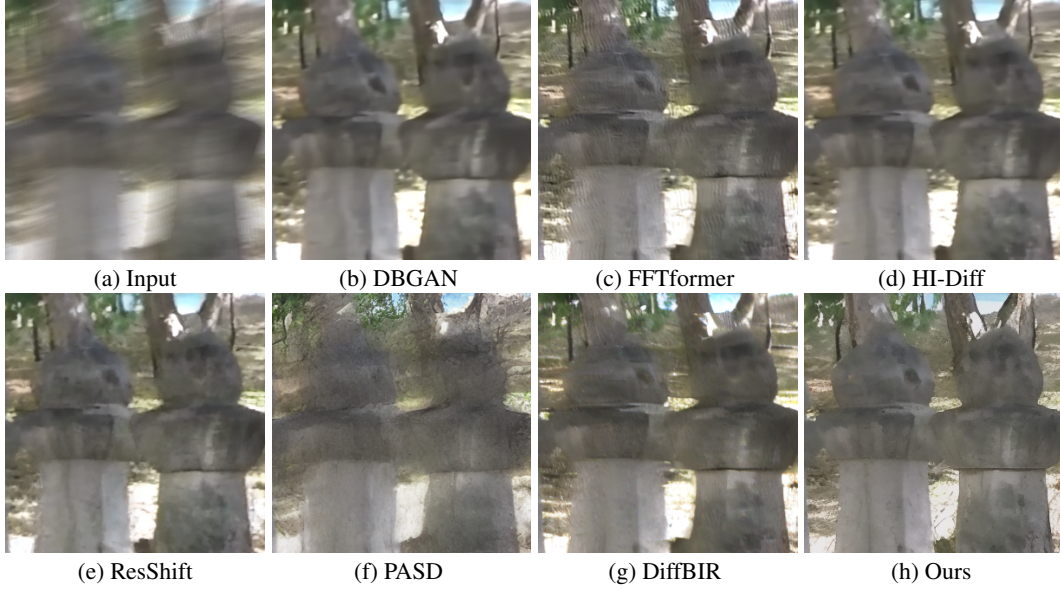


Figure 19: Deblurred results on the GoPro dataset [6]. The results in (b) to (d) do not restore the structures well. In contrast, the proposed method generates a better image with clearer structures.



Figure 20: Deblurred results on the GoPro dataset [6]. The results in (b) to (d) still contain significant blur effects, whereas our method recovers clear structures and details.

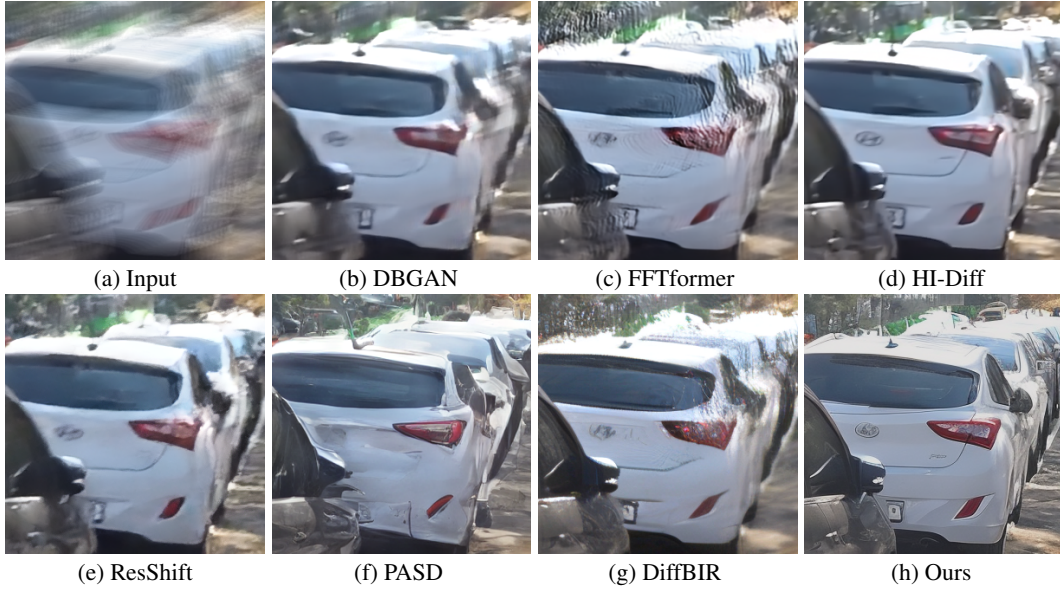


Figure 21: Deblurred results on the GoPro dataset [6]. The deblurred results in (b)-(g) still contain significant blur effects. The proposed method generates a clearer image, where the structures of the cars are much clearer.

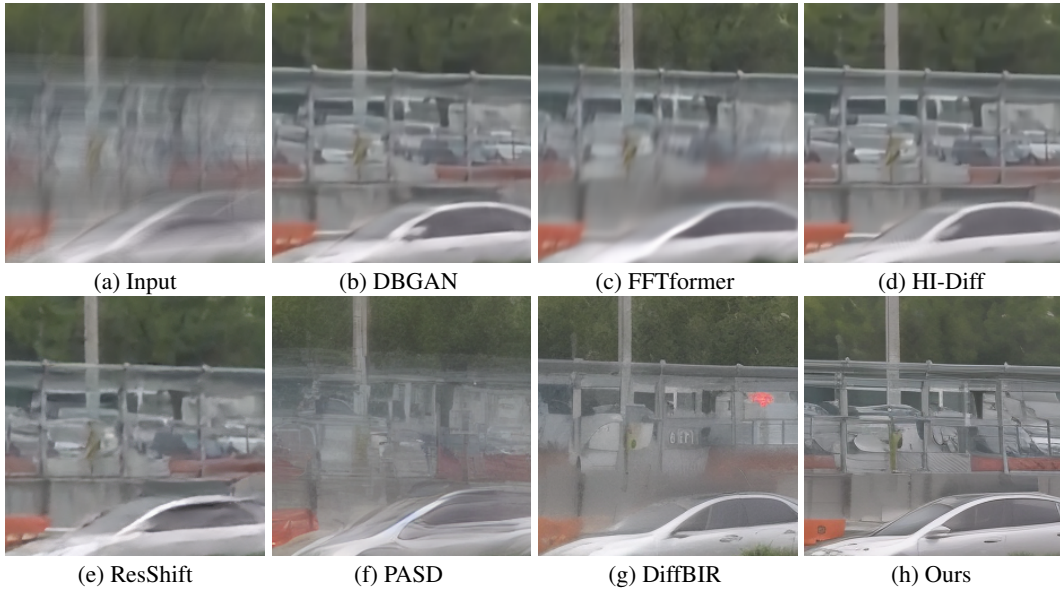


Figure 22: Deblurred results on the GoPro dataset [6]. The results in (b) to (g) do not restore the structures well. In contrast, the proposed method generates a better image with clearer structures.

References

- [1] G. Boracchi and A. Foi. Modeling the performance of image restoration from motion blur. *Image Processing, IEEE Transactions on*, 21(8):3502–3517, aug. 2012.
- [2] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *ECCV*, 2022.
- [3] Sunghyun Cho, Jue Wang, and Seungyong Lee. Video deblurring for hand-held cameras using patch-based synthesis. *ACM Transactions on Graphics (TOG)*, 31:1–9, 2012.
- [4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [5] Xinqi Lin, Jingwen He, Ziyang Chen, Zhaoyang Lyu, Bo Dai, Fanghua Yu, Yu Qiao, Wanli Ouyang, and Chao Dong. Diffbir: Toward blind image restoration with generative diffusion prior. In *ECCV*, 2024.
- [6] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*, 2017.
- [7] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *CVPRW*, 2019.
- [8] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.
- [9] Jaesung Rim, Geonung Kim, Jungeon Kim, Junyong Lee, Seungyong Lee, and Sunghyun Cho. Realistic blur synthesis for learning image deblurring. In *ECCV*, 2022.
- [10] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *CVPR*, 2017.
- [11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.
- [12] Tao Yang, Rongyuan Wu, Peiran Ren, Xuansong Xie, and Lei Zhang. Pixel-aware stable diffusion for realistic image super-resolution and personalized stylization. In *ECCV*, 2024.
- [13] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *CVPR*, 2022.
- [14] Kaihao Zhang, Wenhan Luo, Yiran Zhong, Lin Ma, Björn Stenger, Wei Liu, and Hongdong Li. Deblurring by realistic blurring. In *CVPR*, 2020.
- [15] Kaihao Zhang, Tao Wang, Wenhan Luo, Wenqi Ren, Björn Stenger, Wei Liu, Hongdong Li, and Ming-Hsuan Yang. Mc-blur: A comprehensive benchmark for image deblurring. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(5):3755–3767, 2024.