Supplementary Material for "Towards Efficient and Scale-Robust Ultra-High-Definition Image Demoiréing"

Xin Yu[1], Peng Dai[1], Wenbo Li[2], Lan Ma[3], Jiajun Shen[3], Jia Li[4], and Xiaojuan Qi[1]✉

[1] The University of Hong Kong
[2] The Chinese University of Hong Kong
[3] TCL AI Lab
[4] Sun Yat-sen University

Outline

We supplement the main body of our paper with additional details, discussions, and results in this document. In Section A, we present more details about our dataset capture, which includes a brief analysis of the formation of degraded screen images. In Section B, we provide more implementation details of our network architecture as well as a simple empirical study of loss functions to assist us in selecting a suitable training objective for moiré removal. In Section C, we provide more implementation details of experiments and show more qualitative results and comparisons with other state-of-the-art methods. Furthermore, as shown in Section C.2, we investigate why FHDe^{2}Net fails on this more challenging 4K dataset. We conduct a more detailed discussion of current methods' strategies for handling scale-variation of moiré patterns in Section D.

A Dataset Capture and Analysis

In this section, we first present a brief introduction of the formation of the moiré pattern, and then we provide more details about our capture settings.

A.1 Image Degradation Analysis

The formation of degraded screen images taken with mobile devices can be divided into two processes: the generation of moiré patterns caused by frequency aliasing; and the global color degradation of the image, caused by a series of ISP operations (e.g., auto exposure control, white balance correction, gamma correction, and global tone mapping).

We can model the formation of moiré patterns as a local color unbalanced scaling in the camera's color filter array (CFA). Without loss of generality, consider how one of the green channels in the RGBG raw pattern is collected. As

✉ indicates the corresponding author.

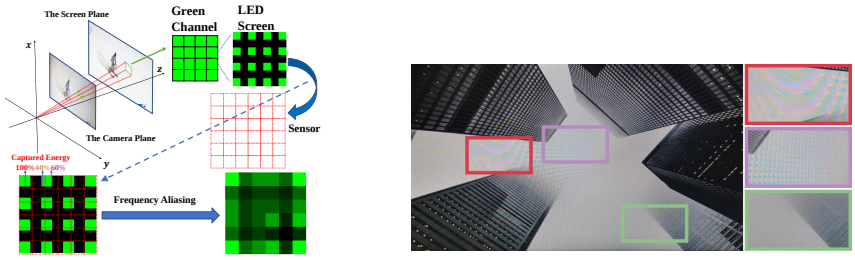

Fig. 1: Left: the formation of the moiré pattern. Notice that there are small gaps between the light-emitting diodes. Right: the characteristics of moiré patterns

shown in Fig. 1, due to a slight misalignment between sensor pixels and LED screen pixels, the energy may shift from one pixel to its neighbors. This flow eventually aligns again after passing a few pixels. Hence, the value of each pixel in this period could be modeled as being multiplied by different scaling factors:

$$\hat{R}(i,j) = R(i,j) * S(i,j), \tag{1}$$

where $\hat{R}(i,j) = \left(\hat{r}_{ij}, \hat{g}^1_{ij}, \hat{b}_{ij}, \hat{g}^2_{ij}\right)$ represents the degraded pixel at the location (i,j) in the Bayer pattern and $R(i,j)$ denotes the clean pixel. $S(i,j) = \left(s^r_{ij}, s^{g_1}_{ij}, s^b_{ij}, s^{g_2}_{ij}\right)$ is the scaling factor for four channels (RGBG) caused by frequency aliasing; $*$ denotes the point-wise multiplication. However, since the LED display and camera Bayer array both emit or receive each channel information in an alternate form, the scaling rules for different channels are not consistent within a cycle. Hence, the camera stores a wrong color distribution, causing the moiré pattern we see.

Furthermore, there is an unavoidable gap when we re-capture an image on the screen. For instance, ambient light can lead to incorrect exposure control, wrong auto white balance, and unnatural tone mapping. Also, corrupted raw data can affect the process of raw image demosaicing. All of these factors contribute to the overall degradation of the color, which can be formulated as:

$$M = F\left(R * S\right), \tag{2}$$

where M is the final degraded screen image and F is a nonlinear function that globally affects the image quality.

Given the above analysis, we could explain the following characteristics (see Fig. 1) of moiré patterns:

Structural distortions: Since the RGB color distributions change in an alternate form, the local illuminance contrasts among the three channels are not consistent. Thus, new structures are created and mixed with original contents.

Diverse degraded forms: In Fig. 1, we show the simplest case of misalignment between two patterns, in which the camera plane and the screen plane are parallel to each other. Obviously, the scaling rule would be quite different if the angle and distance between these two planes were to change, resulting in moiré patterns in

different shapes and scales. This explains why the moiré pattern characteristics highly depend on the geometric relationship between the screen and the camera. **Large-scale patterns in low-frequency regions:** Unlike the natural image captured from real scenes, we capture discrete signals emitted from the LED screen and store them in new discrete forms. Thus, the low-frequency image areas actually become signals with the highest frequency and are more likely to continuously alias with the camera sensor over a long period, resulting in larger moiré patterns.

A.2 More Details about Capture Settings

Based on the above analysis, we thus shoot the screen images via different camera views to produce different patterns and combine multiple devices to produce diverse degradation styles (including pattern appearance and global color style). Specifically, we apply three mobile phones and three digital screens, as shown in Table 1 ($3 \times 3 = 9$ combinations here totally). Notably, the "4K" challenge means the obtained moiré image is at a resolution of ultra-high-definition (i.e., the shooting resolution is 4K). We also compare our dataset with other datasets visually. As seen in Fig. 2, we crop patches from these four datasets at the same resolution 256×256 (the image in TIP2018 dataset [9] is already at a resolution of 256×256). Obviously, compared with other datasets, the image UHDM suffers from more severe moiré artifacts and has less clean image content to harvest in a local window. As a result, it is more challenging for the network to identify the moiré pattern or fill clean content into the degraded region, which has also been demonstrated in [3].

Table 1: The capture devices we apply to get the moiré image

Mobile Phone	Shooting Resolution	Digital Screen	Display Resolution
iPhone XR	4032×3024	LG 27UL650-W	3840×2160
iPhone 13	4032×3024	AOC U2790PQU	3840×2160
Redmi K30 Pro	4624×3472	Philips 243S7EHMB	1920×1080

B Method

In this section, we give details of our network architecture. The overview of our network is shown in Fig. 3. We use skip-connections to connect each level of the encoder and decoder, wherein the features are concatenated.

B.1 Semantic-Aligned Scale-Aware Module (SAM)

As seen in Fig. 3, there are three branches in the pyramid context extraction module wherein the dilated dense block ($L = 5$) is utilized as the backbone

Fig. 2: Comparisons with other datasets; we crop patches from these four datasets at the same resolution 256×256 (the image in TIP2018 dataset [9] is already at a resolution of 256×256)

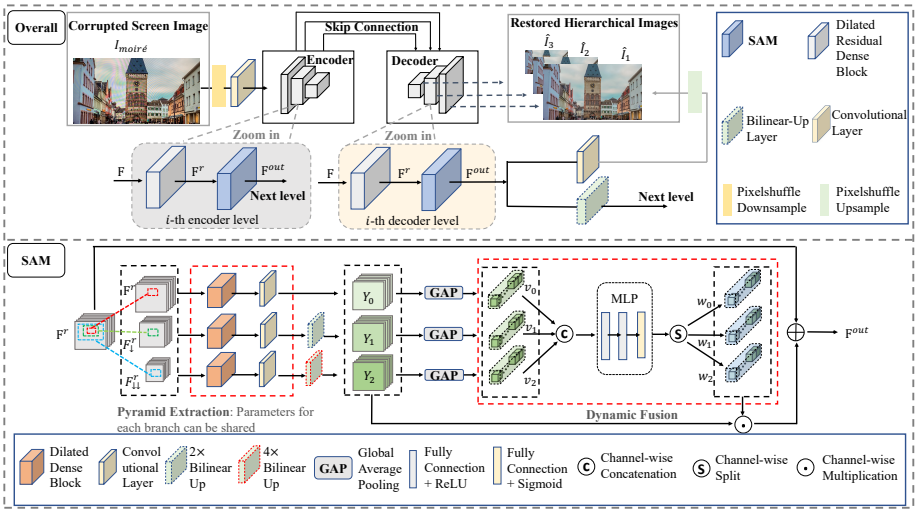

Fig. 3: The pipeline of our ESDNet and the proposed semantic-aligned scale-aware module (SAM)

block to extract the context information. Two bilinear upsampling layers with upsampling ratios 2 and 4 are applied to the second and third branches to align the spatial resolution of the first branch. There are three fully connected layers for the MLP in the cross-scale dynamic fusion module to learn the adaptive weights. We adopt ReLU for the first two layers and Sigmoid for the last layer as our nonlinear activation functions. Specifically, for an input tensor $v \in \mathbb{R}^{1\times1\times3C}$, the channel number is squeezed by a dividing factor 4 in the first layer and then expanded to the original number in the last layer.

Weight-sharing SAM: We apply a weight-sharing strategy for one of our models, denoted as WS-ESDNet, which shares the learnable parameters among the three branches. The WS-ESDNet has fewer parameters while keeping comparable quantitative and qualitative results compared to our standard model ESDNet. The quantitative results have already been shown in Section.5.2 in the main body of our paper, and qualitative results are illustrated in Section C. This demonstrates that the performance gain primarily benefits from our architecture design rather than increased model parameters.

Table 2: Detail of the encoder; DRDB denotes the dilated residual dense block consisting of three convolution layers

Level	Block Type	Input Channels	Output Channels	Inter Channels	Dilation Rates
1	Pixelshuffle downsampling	3	12	-	-
	5×5 Conv + ReLU	12	48	-	-
	DRDB	48	48	32	(1, 2, 1)
	SAM	48	48	32	(1, 2, 3, 2, 1)
2	Stride=2, 3×3 Conv	48	96	-	-
	DRDB	96	96	32	(1, 2, 1)
	SAM	96	96	32	(1, 2, 3, 2, 1)
3	Stride=2, 3×3 Conv	96	192	-	-
	DRDB	192	192	32	(1, 2, 1)
	SAM	192	192	32	(1, 2, 3, 2, 1)

B.2 Empirical Study of Loss Functions

The loss function plays an essential role in guiding model updates and encouraging the model to learn natural patterns from data. To this end, we carry out an empirical study to investigate the impacts of different loss functions on image demoiréing.

We evaluate traditional L_1 loss and its combination with perceptual losses [4] where the features are respectively from the end of block_1, block_2, block_3, block_4 and block_5 of a pre-trained VGG-16 network [8]. We develop a simple task to study the effectiveness of these loss functions on removing undesirable moiré patterns. Specifically, we choose a degraded screen image M with severe structural distortions and its corresponding clean ground-truth I; our aim is to restore M by optimizing $\theta^\star = \arg\min_\theta D(I, f_\theta(M))$ through our designed network f_θ, where D denotes the loss function, and $\hat{I} = f_{\theta^\star}(M)$ is the recovered image. As shown in Fig. 4, the single L_1 loss or its combination with the

Table 3: Detail of the decoder; DRDB denotes the dilated residual dense block consisting of three convolution layers

Level	Block Type	Input Channels	Output Channels	Inter Channels	Dilation Rates
3	3×3 Conv + ReLU	192	64	-	-
	DRDB	64	64	32	(1, 2, 1)
	SAM	64	64	32	(1, 2, 3, 2, 1)
Output Layer	3×3 Conv	64	12	-	-
	Pixelshuffle upsampling	12	3	-	-
Transition Layer	Bilinear-Up Layer	64	64	-	-
2	3×3 Conv + ReLU	160	64	-	-
	DRDB	64	64	32	(1, 2, 1)
	SAM	64	64	32	(1, 2, 3, 2, 1)
Output Layer	3×3 Conv	64	12	-	-
	Pixelshuffle upsampling	12	3	-	-
Transition Layer	Bilinear-Up Layer	64	64	-	-
1	3×3 Conv + ReLU	112	64	-	-
	DRDB	64	64	32	(1, 2, 1)
	SAM	64	64	32	(1, 2, 3, 2, 1)
Output Layer	3×3 Conv	64	12	-	-
	Pixelshuffle upsampling	12	3	-	-
Transition Layer	Bilinear-Up Layer	64	64	-	-

shallow block_1 perceptual loss cannot guide the network to remove unnecessary structures; they are effective in restoring the pixel-level color due to their low-level nature. Meanwhile, the loss functions derived from block_4 and block_5 features, containing too deep semantic-level information, will lead the predicted image to lose its textures. In contrast, perceptual loss with features from block_2 and block_3 can encourage the network to remove undesirable structures while preserving the original texture, a good signal for image demoiréing. In particular, the model trained with block_3 recovers more details with satisfying local contrasts. Hence, the block_3 might be the most suitable layer to construct the training objective.

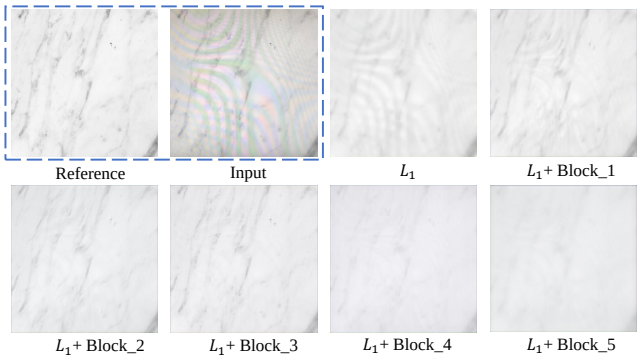

Fig. 4: The optimal results by fitting different loss functions for a single moiré image

Although many previous works [2,3,6] have already adopted the perceptual loss as a regularization term, they often overlook the importance of precisely choosing a suitable layer for this specific task, which is crucial, as different features will encourage the network to optimize the network in different directions.

C Experiments

C.1 Implementation Details

We implement all the experiments using PyTorch on an NVIDIA RTX 3090 GPU card. The learning rate is initially set to 0.0002 and scheduled by cyclic cosine annealing [7], and models are optimized by Adam [5] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For UHDM dataset, we set the batch size as 2. Notably, we conduct benchmark implementations of other methods [9,13,2,3,1,6] on our dataset sufficiently. For DMCNN [9], MDDM [1], WDNet [6] and MBCNN [13], we randomly crop a 768×768 patch from the ultra-high-definition images, and train the model for 150 epochs, i.e., the totally same setting with ours. For FHDe^{2}Net [3], due to its different multi-stage nature and high computational cost, we can only follow its default setting in the official released code for training (i.e., down-sampled-resolution 384×384 for training its global stage and cropped 384×384 region for training the following three cascaded networks). For MopNet[2], we freeze its pre-trained classification sub-network and train its edge-prediction sub-network and demoiréing sub-network for 150 epochs, wherein we also crop a 384×384 region for training. During inference, since MopNet cannot directly process the 4K image due to its heavy memory cost, we downsample the input image into 1080p (the highest resolution it can process on a single GPU) resolution and then upsample the result back to 4K resolution.

Other datasets: For FHDMi [3] and LCDmoiré [10] dataset, we randomly crop a 512×512 patch from the high-definition images, and train the model for 150 epochs with the batch size as 2. For TIP2018 dataset [9], we follow the benchmark setting, i.e., we first resize the image into a 286×286 resolution and then do center crop to produce a 256×256 resolution image for both training and testing. We train our models for 70 epochs and set batch size to 4.

C.2 Discussion about FHDe^{2}Net

We find that in the new dataset UHDM, FHDe^{2}Net suffers from a more significant performance drop than other methods. To this end, we conduct a parameters searching and analysis. Specifically, since we find the key challenge is to fuse the high-frequency detail, we mainly analyze the training of the last stage, i.e., the FDN and FRN (please refer to [3] for more details). Since the learning rate is scheduled by cyclic cosine annealing, which warms up every 50 epochs, we evaluate the performances after the FDN and FRN (the last stage of FHDe^{2}Net) have been trained for 50, 100, and 150 epochs, respectively. As shown in Table 4, with the increase of training time, SSIM improves significantly, but LPIPS degrades

simultaneously. For this phenomenon, we attribute the reasons to two aspects, as elaborated upon below.

On the one hand, current low-level metrics have several limitations and cannot fully measure the demoiréing performance (see Fig. 5). For example, PSNR is a pixel-wise metric sensitive to pixel misalignment and slight color shift, which has limited effect in measuring the structural distortion caused by the moiré pattern. SSIM is more robust to evaluate structural distortion yet still sensitive to the unstructured distortion (e.g., pixel shift, rotation.), which is unavoidable in real-world data pairs. LPIPS has been proven to be more consistent with human perception; however, it is sensitive to blur as demonstrated in [11].

Fig. 5: Current metrics have some limitations. In this case, FHDe^{2}Net removes the moiré pattern more cleanly yet is still behind the MBCNN if evaluated by the three metrics

Fig. 6: Comparisons between the result produced by global demoiréing stage and the final result (i.e., "Detail Refine"), in which the PSNR is almost unchanged while LPIPS achieves significant improvement

On the other hand, this indicates FHDe^{2}Net has reached its limit in making the trade-off between large-scale moiré removal and high-frequency details preservation. To explore whether this stage plays a role in high-frequency detail

recovery, we compare it with the initial low-resolution result produced by the global demoiréing stage. As shown in Table 4, compared with the initial result (i.e., "Low-resolution"), the fine-tuned model (i.e., "150 epoch") achieves a significant improvement in LPIPS which indicates the detail has been recovered to some degree (but not been fully recovered, see Fig. 6). However, the PSNR is almost unchanged, indicating that this stage may not work well for color recovery. One possible reason is that the fusion stage only utilizes the Y-channel's information of the original high-resolution image but lacks UV-channels' high-resolution information. Besides, to avoid the effect of pixel misalignment, FHDe^{2}Net does not adopt pixel-wise loss terms (e.g., L_1, L_2), which may prevent it from recovering the global color style. Under this circumstance, the accurate color information loses significantly, negatively affecting all three metrics, especially for the PSNR.

Table 4: Quantitative results of different implementations of FHDe^{2}Net on UHDM dataset. "Pre-train" denotes the inference result by directly applying the official released pre-train model on FHDMi dataset [3], "Low-resolution" denotes the intermediate result produced by the first global demoiréing stage in FHDe^{2}Net

Metrics	Input	Pre-train	Low-resolution	50 epoch	100 epoch	150 epoch
PSNR↑	17.117	18.052	20.333	20.312	20.313	20.338
SSIM↑	0.5089	0.5986	0.7408	0.7290	0.7365	0.7496
LPIPS↓	0.5314	0.4929	0.4669	0.3397	0.3429	0.3519

In fact, we have conducted several parameters searching for the last stage's training (consists of two sub-networks FDN and FRN), trying to improve the performance of FHDe^{2}Net. To be precise, we adjust the loss weights to guide the networks' optimization. As illustrated in Eq.(3), the overall loss function of the last stage consists of two parts: L_{FDN} and L_{FRN}, where L_{FDN} aims to reconstruct the high-resolution gray-scale image (i.e., the Y-channel of YUV color space) and L_{FRN} aims to further fuse the color information (more details can be referred to [3]):

$$\mathcal{L}_{\text{last}}(I, \hat{I}) = \mathcal{L}_{\text{FDN}}(I_Y, \hat{I}_Y) + \lambda \times \mathcal{L}_{\text{FRN}}(I, \hat{I}) \tag{3}$$

where I is the ground-truth and $\hat{I}$ is the network's output, I_Y and $\hat{I}_Y$ denote their Y-channel components, respectively. Moreover, for L_{FRN}, it is essentially a CoBi [12] loss, which aims to measure the similarity between unaligned image pairs, consisting of a term $\mathbb{D}$ to measure feature similarity and a term $\mathbb{D}'$ to compute the spatial distance between these two pixels (with a weight w_s), i.e.,:

$$\mathcal{L}_{\text{FRN}}(\hat{I}, I) = \frac{1}{N} \sum_{i}^{N} \min_{j=1,\ldots,M} \left((1 - w_s)\mathbb{D}\left(p_i, q_j\right) + w_s \mathbb{D}'\left(p_i, q_j\right) \right) \tag{4}$$

where p_i, q_j stand for the feature vectors from the output image $\hat{I}$ and clean image I at the spatial position indexed by i and j, respectively. N, M denote the amounts of features (i.e., the amounts in searching space).

We try several (λ, w_s) combinations to train the model. For fast exploration, we train every model for 50 epochs and compare their results, as shown in Table 5. However, since the metrics' changes of each model are not significant, we use the default parameter settings to report the results in our main paper.

In summary, although FHDe^{2}Net achieves the best (except for ours) result on the FHDMi dataset [3], this framework is not robust under the higher resolution setting. Moreover, its complex module designs further render it hard to be applied to the 4K scenario due to unacceptable increased computational costs.

Table 5: Quantitative comparisons of different weights for training FHDe^{2}Net. "A" denotes the default model where $(\lambda, w_s) = (1, 0.5)$; "B" denotes $(\lambda, w_s) = (0.5, 0.5)$; "C" denotes $(\lambda, w_s) = (2, 0.5)$; "D" denotes $(\lambda, w_s) = (1, 0.7)$; "E" denotes $(\lambda, w_s) = (1, 0.2)$

Metrics	Input	Pre-train	Model A	Model B	Model C	Model D	Model E
PSNR↑	17.117	18.052	20.312	20.282	20.174	20.251	19.050
SSIM↑	0.5089	0.5986	0.7290	0.7392	0.7350	0.7435	0.7240
LPIPS↓	0.5314	0.4929	0.3397	0.3409	0.3359	0.3497	0.3566

C.3 SAM for Other Methods

We demonstrate that equipping with the proposed SAM can also help other methods to achieve performance gain. Here we conduct experiments on MDDM [1], DMCNN [9] and MBCNN [13], where we stack SAM in these networks. As shown in Table 6, all metrics have improvements.

Table 6: Effects of the proposed SAM. We add our SAM to current methods DMCNN [9], MDDM [1] and MBCNN [13] to improve their performances

Metrics	Input	DMCNN/(+SAM)	MDDM/(+SAM)	MBCNN/(+SAM)
PSNR↑	17.117	19.914/**20.769**	20.088/**20.883**	21.414/**21.532**
SSIM↑	0.5089	0.7575/**0.7699**	0.7441/**0.7640**	0.7932/**0.7940**
LPIPS↓	0.5314	0.3764/**0.3630**	0.3409/**0.3299**	0.3318/**0.3302**

C.4 More Qualitative Comparisons

As seen in Fig. 8-15, we provide more visual results and comparisons with current state-of-the-art methods on three real-world demoiréing datasets: UHDM (resolution: 3840×2160), FHDMi [3] (resolution: 1920×1080) and TIP2018 [9] (resolution: 256×256). Apparently, our model can remove moiré patterns more cleanly and preserve high-frequency details better.

D Revisit Current Multi-Scale Schemes in Image Demoiréing

We have discussed in our main paper that a key challenge in image demoiréing is the scale variation of the moiré pattern. In this section, we conduct a more detailed analysis of multi-scale schemes in current demoiréing works. As shown in Fig. 7, we summarize these schemes into two parts: single-stage training and multi-stage training. We figure out their inefficiency and insufficiency, which limit their performance when processing ultra-high-definition images.

D.1 Single-Stage Training

Most of the demoiréing works adopt a single-stage framework, i.e., given a moiré image $I_{\text{moiré}} \in \mathbb{R}^{h\times w\times 3}$, an end-to-end network $\mathbf{F}$ is trained to produce the final demoiréd image $I_{\text{demoiré}}$:

$$I_{\text{demoiré}} = \mathbf{F}(I_{\text{moiré}}) \tag{5}$$

Specifically, they embed different multi-scale schemes into their networks, which can be simplified and summarized into two topological architectures: parallel multi-scale and cascaded multi-scale.

Cascaded multi-scale: Adopted by MopNet[2], MBCNN[13] and WDNet[6] (Note that although MopNet is a multi-stage framework, it harvests multi-scale information in one sub-network), the insight in cascaded multi-scale strategy is utilizing features from different-depth layers to get multi-scale representations. As shown in the right upper part of Fig. 7, the moiré image first goes through an encoder that contains three levels to extract features. Then the intermediate results in each level are fused together and fed to the decoder for reconstruction. Since features are produced in different-depth layers, their receptive fields are different (the receptive field is larger for a deeper feature). However, another fact is ignored: features at different depths have different semantic meanings. For example, features extracted in the early layer usually contain low-level information such as edge, while features in the deeper layers contain more abstract attributes learned by the network. Recall that the scale-variation challenge means that the observed object remains the same for all attributes (e.g., color, shape) except for the scale that appeared in an image (i.e., pixels it counts). Thus, a more reasonable design is the network can extract multi-scale information at the same semantic level (i.e., depth level). Further, a robust network should harvest multi-scale information at each semantic level to handle different attributes. Based on this analysis, we find that this cascaded strategy lacks multi-scale ability at a specific semantic level, limiting its scale-robust ability.

Parallel multi-scale: The parallel multi-scale indicates construction of parallel high-resolution to low-resolution branches to process different-scale features, as adopted in DMCNN[9] and MDDM[1]. At each scale, several convolutional blocks are stacked to extract features and finally produce a three-channel output. Without loss of generality, we suppose there are three scales and three convolutional blocks in each scale to illustrate and analyze this strategy.

As shown in the left upper part of Fig. 7, the moiré image first goes through several downsampling convolutional heads with different strides to obtain shallow representations with different resolutions:

$$J_i = \mathrm{Conv}_i(I_{\text{moiré}}), i = 1, 2, 3 \tag{6}$$

where Conv_i denotes convolutional block with stride $s = 2^{i-1}$, $J_i \in \mathbb{R}^{\frac{h}{2^{i-1}} \times \frac{w}{2^{i-1}} \times c}$. After that, each J_i is fed to several convolutional blocks in parallel:

$$X_i = F_i^3(F_i^2(F_i^1(J_i))), i = 1, 2, 3 \tag{7}$$

where F_i^j denotes the j-th blocks in i-th scale (branch), $X_i \in \mathbb{R}^{\frac{h}{2^{i-1}} \times \frac{w}{2^{i-1}} \times 3}$. Then an upsampling layer would be utilized to align the spatial size of each-scale outputs, followed by a summation operation to get the final prediction $I_{\text{demoiré}}$:

$$I_{\text{demoiré}} = X_1 + X_{2\uparrow} + X_{3\uparrow\uparrow} \tag{8}$$

Unlike the cascaded multi-scale scheme, the insight here is to reduce the resolution at the input stage, so different branches have different receptive fields. However, the problem is, this framework only fuses the results at the end of each branch, ignoring the interaction of the intermediate features. As a result, each extracted feature is only determined by its current branch (scale), dramatically limiting the network's representation ability. For example, to produce the feature F_2^2, the network only utilizes the information from F_2^1. However, a more representative feature needs to harvest multi-scale information from last semantic level. Only fusing information in the last layer results in coarse moiré pattern removal, as shown in Fig. 8-15.

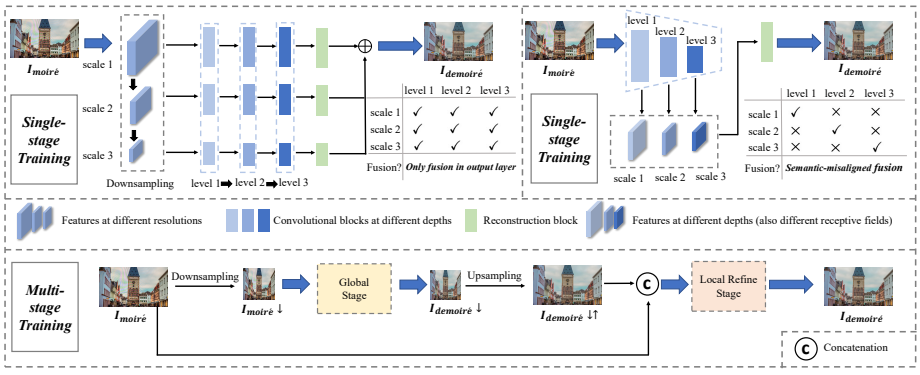

Fig. 7: A summary of current works for solving the multi-scale challenge in image demoiréing

D.2 Multi-Stage Training

FHDe^{2}Net [3] is the only current work which proposes to tackle real-world high-definition moiré images. Due to the increased resolution, the scale of the moiré pattern would expand extremely larger, which has been the main challenge in the high-definition demoiréing. The central insight in this work is adopting a multi-stage framework to handle this problem, the networks of which are trained step by step. As shown in the lower part of Fig. 7, the overall framework can be divided into two stages: the global stage and the local refine stage (In fact, it consists of four sub-networks, but we summarize it into two stages here for analysis). The input of the global stage is a downsampled low-resolution (384×384) moiré image, so the network in this stage can obtain a full-image-size receptive field. Although the large-scale moiré pattern can be removed, the images' high-frequency details are severely lost due to the downsampling operation. Hence, in the local refinement stage, the original high-resolution image would be utilized to guide the low-resolution demoiréd image to recover the details. However, our experiments find it hard for the network to differentiate the moiré pattern from the image textures, leading to the reintroduction of the moiré pattern and unsatisfactory texture recovery. Furthermore, its internal complex module design shows a heavy computational burden, which is unacceptable for ultra-high-definition image demoiréing.

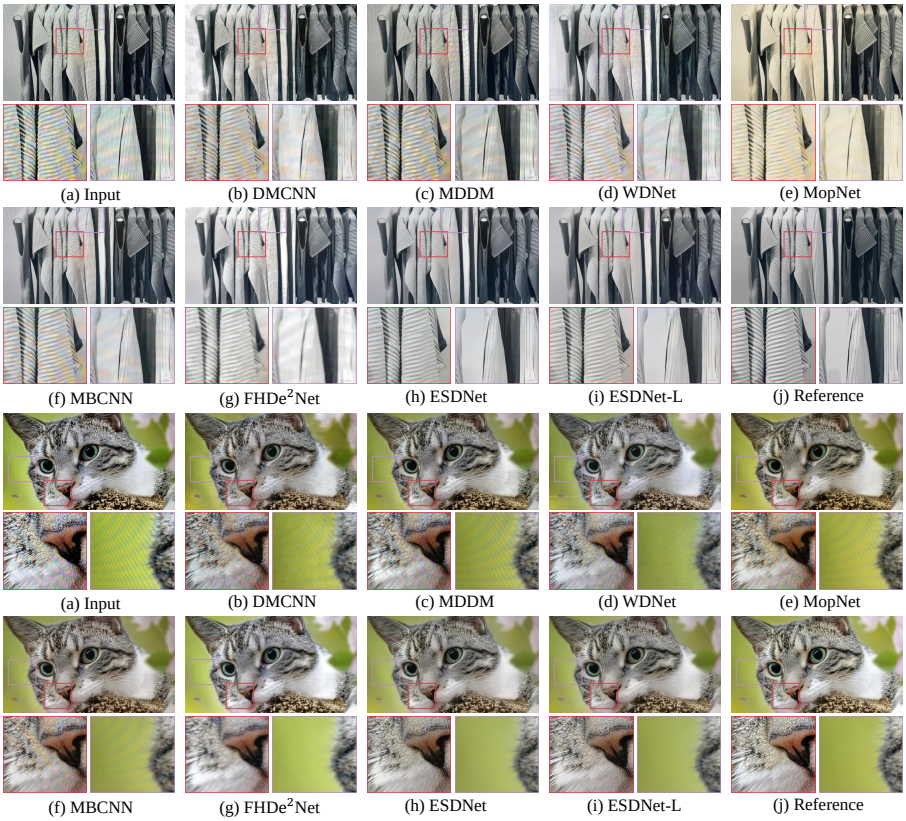

Fig. 8: Qualitative comparisons of our models with other state-of-the-art methods on the UHDM dataset, ESDNet is our standard model and ESDNet-L is our larger model

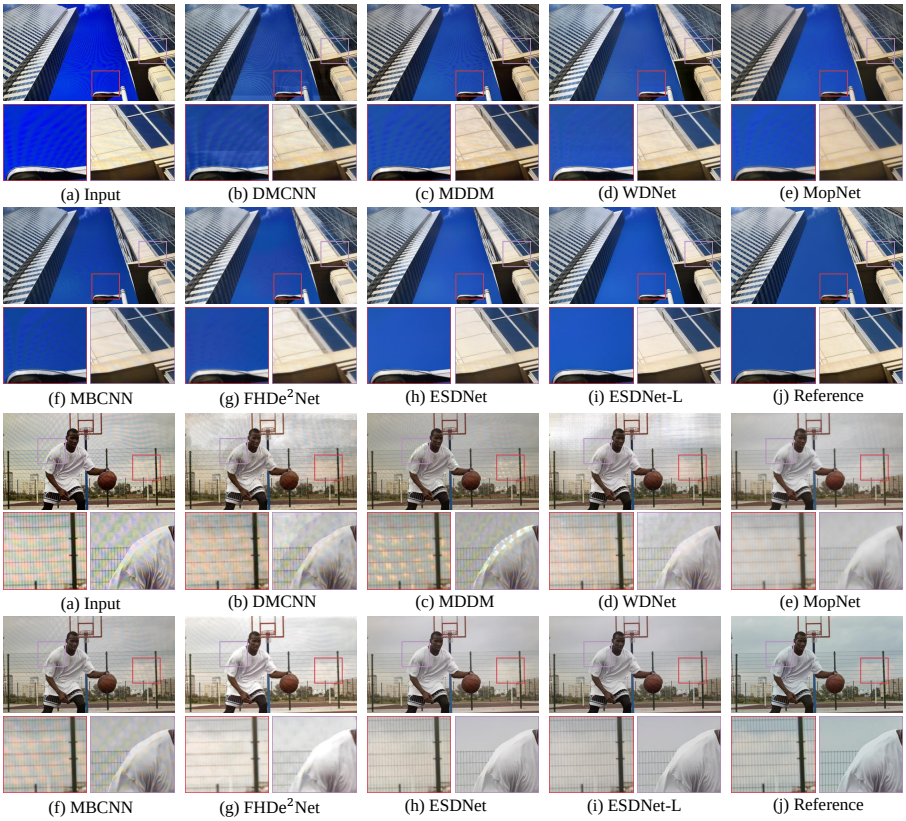

Fig. 9: Qualitative comparisons of our models with other state-of-the-art methods on the UHDM dataset, ESDNet is our standard model and ESDNet-L is our larger model

Fig. 10: Qualitative comparisons of our models with other state-of-the-art methods on the UHDM dataset, ESDNet is our standard model and ESDNet-L is our larger model

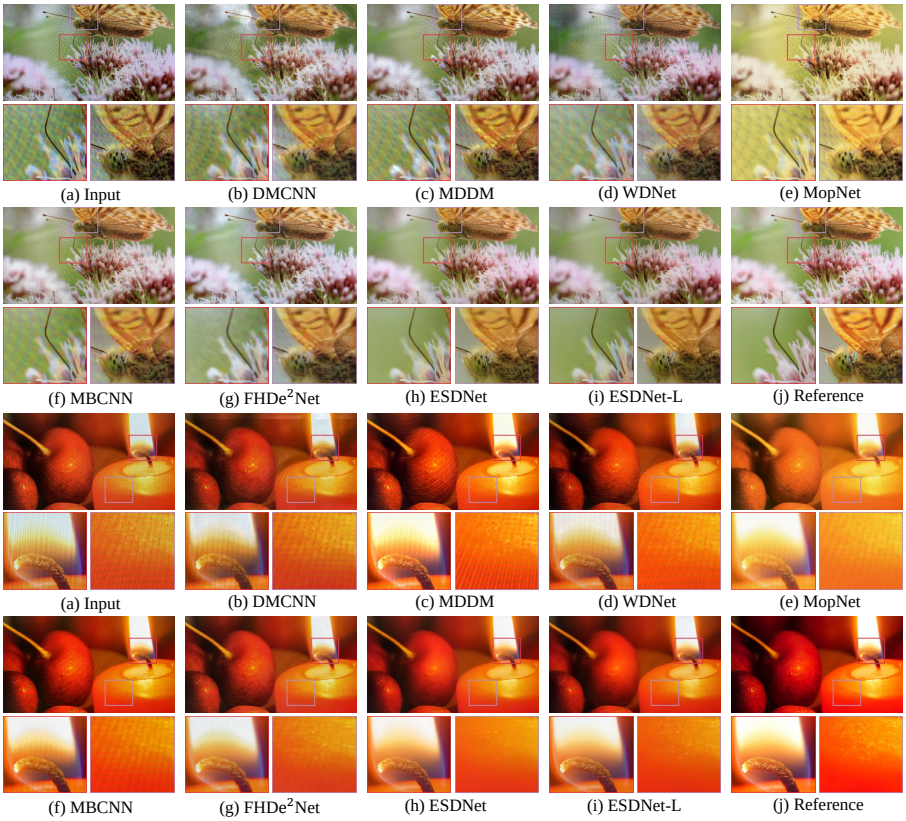

Fig. 11: Qualitative comparisons of our models with other state-of-the-art methods on the UHDM dataset, ESDNet is our standard model and ESDNet-L is our larger model

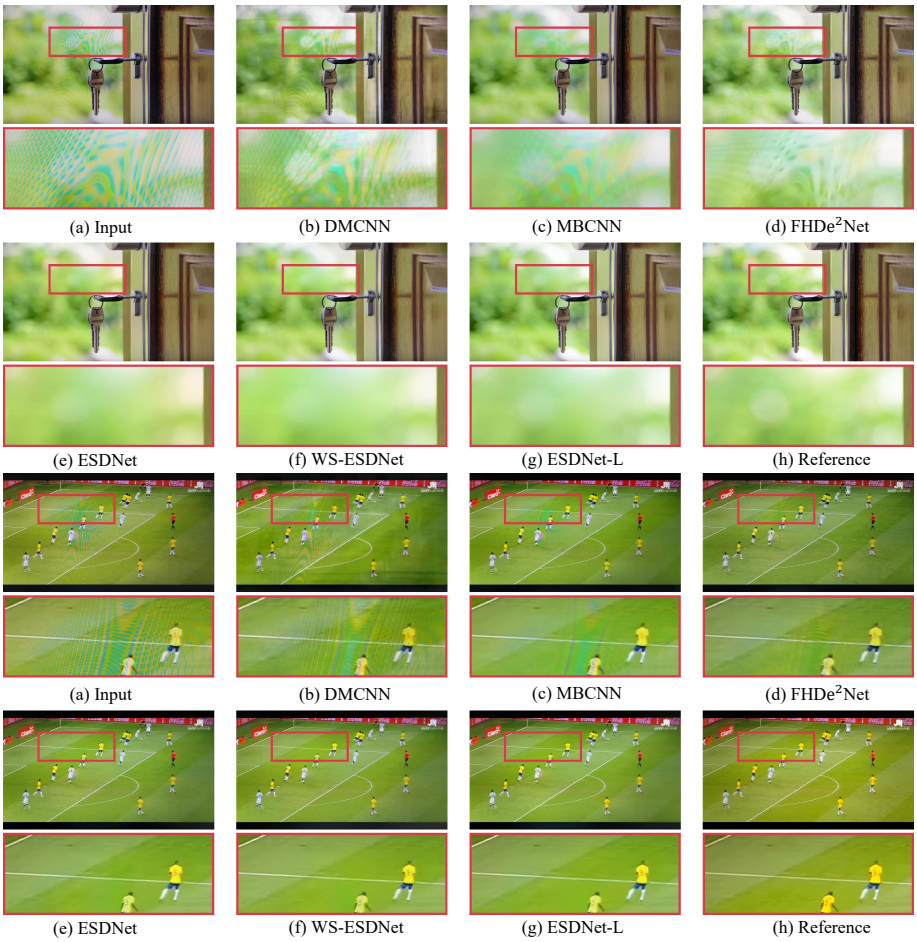

Fig. 12: Qualitative comparisons of our models with three representative state-of-the-art methods on the FHDMi dataset [3], including DMCNN [9], MBCNN [13] and FHDe^{2}Net [3]. ESDNet is our standard model and ESDNet-L is our larger model. WS-ESDNet is our more lightweight model, the parameters of which are shared in three branches of pyramid context extraction module

Fig. 13: Qualitative comparisons of our models with three representative state-of-the-art methods on the FHDMi dataset [3], including DMCNN [3], MBCNN [13] and FHDe^{2}Net [3]. ESDNet is our standard model and ESDNet-L is our larger model. WS-ESDNet is our more lightweight model, the parameters of which are shared in three branches of pyramid context extraction module

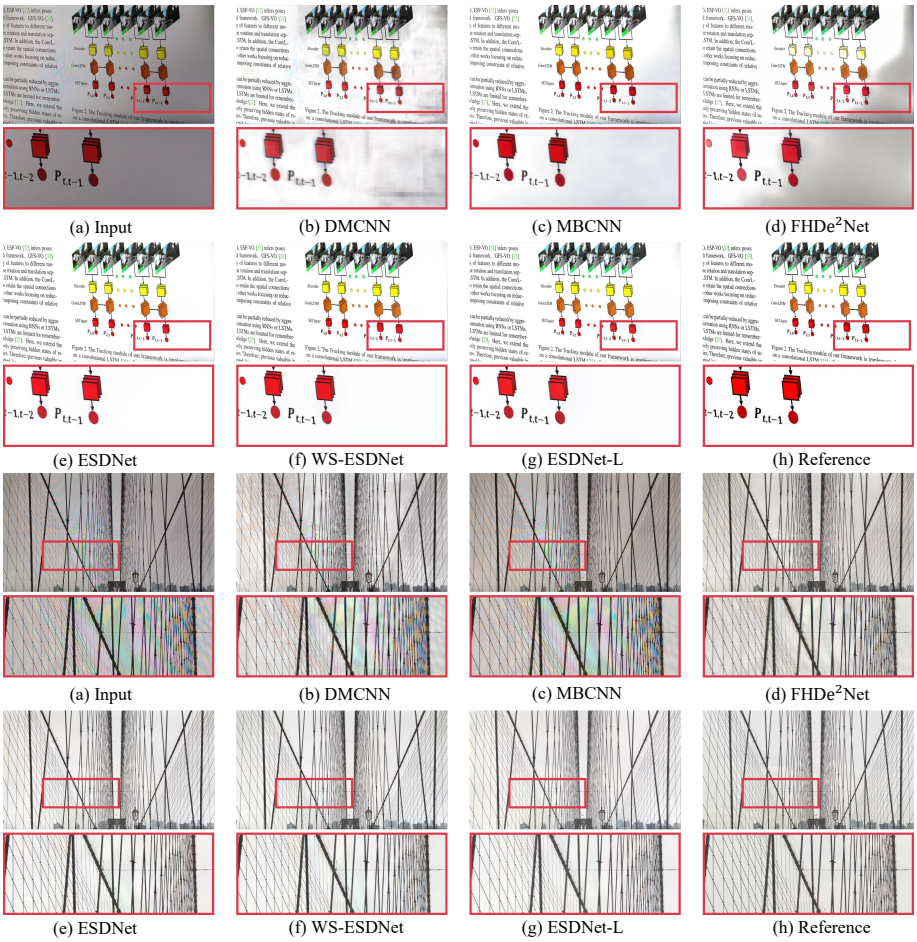

Fig. 14: Qualitative comparisons of our models with three representative state-of-the-art methods on the FHDMi dataset [3], including DMCNN [9], MBCNN [13] and FHDe^{2}Net [3]. ESDNet is our standard model and ESDNet-L is our larger model. WS-ESDNet is our more lightweight model, the parameters of which are shared in three branches of pyramid context extraction module

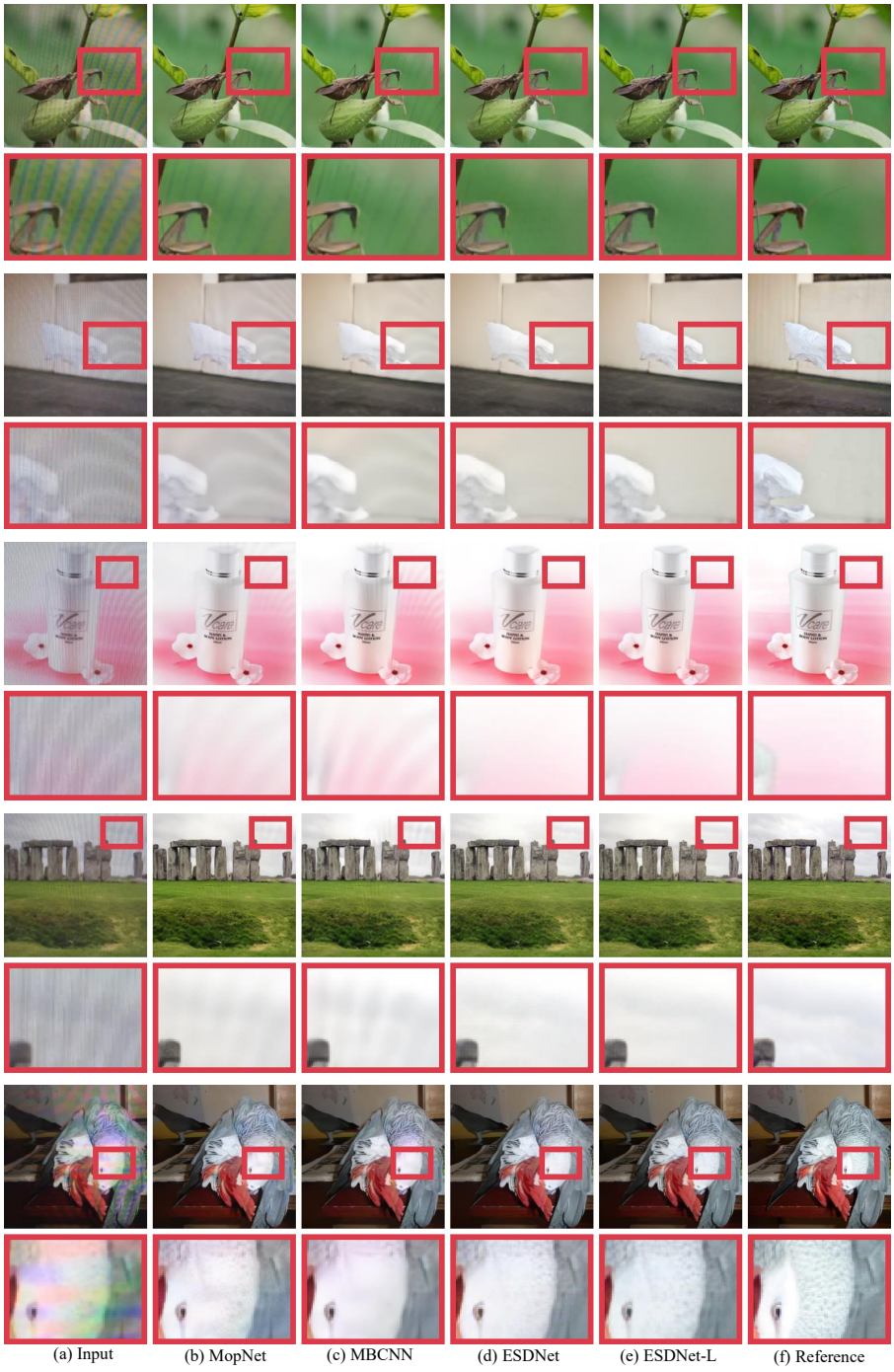

Fig. 15: Qualitative comparisons of our models with two representative state-of-the-art methods on the TIP2018 dataset [9], including MopNet [2] and MBCNN [13]. ESDNet is our standard model and ESDNet-L is our larger model

References

1. Cheng, X., Fu, Z., Yang, J.: Multi-scale dynamic feature encoding network for image demoiréing. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 3486–3493. IEEE (2019) 7, 10, 11
2. He, B., Wang, C., Shi, B., Duan, L.Y.: Mop moire patterns using mopnet. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2424–2432 (2019) 7, 11, 21
3. He, B., Wang, C., Shi, B., Duan, L.Y.: Fhde 2 net: Full high definition demoireing network. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16. pp. 713–729. Springer (2020) 3, 7, 9, 10, 13, 18, 19, 20
4. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European conference on computer vision. pp. 694–711. Springer (2016) 5
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 7
6. Liu, L., Liu, J., Yuan, S., Slabaugh, G., Leonardis, A., Zhou, W., Tian, Q.: Wavelet-based dual-branch network for image demoiréing. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16. pp. 86–102. Springer (2020) 7, 11
7. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016) 7
8. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) 5
9. Sun, Y., Yu, Y., Wang, W.: Moiré photo restoration using multiresolution convolutional neural networks. IEEE Transactions on Image Processing **27**(8), 4160–4172 (2018) 3, 4, 7, 10, 11, 18, 20, 21
10. Yuan, S., Timofte, R., Slabaugh, G., Leonardis, A., Zheng, B., Ye, X., Tian, X., Chen, Y., Cheng, X., Fu, Z., et al.: Aim 2019 challenge on image demoireing: Methods and results. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 3534–3545. IEEE (2019) 7
11. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018) 8
12. Zhang, X., Chen, Q., Ng, R., Koltun, V.: Zoom to learn, learn to zoom. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3762–3770 (2019) 9
13. Zheng, B., Yuan, S., Slabaugh, G., Leonardis, A.: Image demoireing with learnable bandpass filters. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3636–3645 (2020) 7, 10, 11, 18, 19, 20, 21