Fast and Efficient Restoration of Extremely Dark Light Fields

Mohit Lamba  
Indian Institute of Technology Madras  
Department of Electrical Engineering  
eel8d009@small.iitm.ac.in

Kaushik Mitra  
Indian Institute of Technology Madras  
Department of Electrical Engineering  
kmitra@ee.iitm.ac.in

Abstract

The ability of Light Field (LF) cameras to capture the 3D geometry of a scene in a single photographic exposure has become central to several applications ranging from passive depth estimation to post-capture refocusing and view synthesis. But these LF applications break down in extreme low-light conditions due to excessive noise and poor image photometry. Existing low-light restoration techniques are inappropriate because they either do not leverage LF’s multi-view perspective or have enormous time and memory complexity. We propose a three-stage network that is simultaneously fast and accurate for real world applications. Our accuracy comes from the fact that our three stage architecture utilizes global, local and view-specific information present in low-light LFs and fuse them using an RNN inspired feedforward network. We are fast because we restore multiple views simultaneously and so require less number of forward passes. Besides these advantages, our network is flexible enough to restore a $m \times m$ LF during inference even if trained for a smaller $n \times n$ ($n < m$) LF without any finetuning. Extensive experiments on real low-light LF demonstrate that compared to the current state-of-the-art, our model can achieve up to 1 dB higher restoration PSNR, with $9 \times$ speedup, $23\%$ smaller model size and about $5 \times$ lower floating-point operations.

1. Introduction

Unlike conventional cameras, a lenslet-based Light Field (LF) camera [1, 37, 30, 12] captures multiple views, called Sub-Aperture-Images (SAIs), of a scene in a single exposure. This implicit method of capturing a scene’s 3D structure has enabled a wide range of applications such as post-capture refocusing & aperture control [37, 36], depth estimation [20, 52, 48], structure from motion [38], augmented reality [18], and autonomous driving [5]. However, in low light, such as night-time, the SAIs are heavily corrupted by photon noise and contain inadequate color information. This prohibits performing any feature correspondences or satisfying photometric constraints across SAIs, rendering the captured LF useless for downstream applications. Thus it is crucial to design a method for low-light LF restoration.

The existing low-light enhancement techniques are mostly designed for single-frame images [3, 4, 13, 14] that do not utilize the rich LF information. Consequently, their restorations tend to be blurry or noisy. Very recently, a few LF based methods such as L3Fnet [26] and MTO [58] have been proposed to alleviate this problem to a reasonable extent. However, as shown in Fig.1, they have an enormous time-memory complexity that is prohibitive for a real world deployment. For example, even on a high-end CPU the existing LF based methods take 5 – 10 minutes to restore a single $9 \times 9$ LF. Our goal is to design a much faster and memory efficient solution with possibly better restoration quality.

To better capture complementary information present in different LF views our model, as shown in Fig. 2, consists
Figure 2. Our three-stage network for restoring light fields captured in extreme low-light conditions.

of three stages: Stage-I looks at all the views to compute a global embedding. Stage-II output is view-specific and Stage-III gives exclusive attention to the local neighborhood of LF views. Finally, our RNN inspired feedforward network uses all three complementary information to restore LF views.

In Stage-III, we use a RNN inspired feedforward network and not a standard U-net [42]. The immensely successful U-net architecture was originally designed for single-frame images like those obtained from conventional DSLR and smartphone cameras. But we observed that it lacks the expressiveness to capture long-range dependencies between various LF views, especially if the number of views is large. In contrast, RNNs are specially designed for long-range sequence modeling and past methods such as LFNet [53] used RNNs for super-resolving LF views. However, compared to feedforward architectures such as U-nets, RNNs have lower inference speed and more susceptible to vanishing gradients. To thus obtain better restoration quality and faster inference, we analyze the multi-scale processing of a U-net architecture in a RNN framework and, consequently, develop a novel feedforward network by unfolding RNNs in time.

Stage-I and Stage-III of our network share the weights across all SAIs and thus lose the sense of discriminating between SAIs while processing a SAI and its neighborhood. However, differentiating between SAIs is very important because each SAI is captured from a different portion of the main camera lens, leading to different characteristics and distortions across SAIs. We thus introduce Stage-II in between Stage-I and Stage-III, which uses separate network weights for different SAIs. Like ResLF [59], we could have used different weights for each SAI throughout the network but at the expense of having a prohibitively huge model size, inappropriate for real-world deployment.

Another interesting feature of Stage-II is that instead of directly learning the weights for each SAI during training, it instead learns the parameters for a series of fully connected layers, which can estimate the convolutional weights for different SAI based on their angular coordinates $(s, t)$. This allows the network to restore a $m \times m$ LF during inference, even if it is trained for a smaller $n \times n$, $(n < m)$ LF.

Almost all neural network based LF methods require $n^2$ forward passes to process a $n \times n$ LF, which is the main reason for extremely slow inference. Recognizing this fact, we restore multiple SAIs in a single forward pass to substantially reduce the total number of forward passes required to restore low-light LF.

Our contributions: 1) We use a three-stage architecture to utilize global, local and view specific information present in low-light LF and fuse them using a novel RNN inspired feedforward network for superior restoration. 2) Our model can restore a $m \times m$ LF, even if trained for a smaller $n \times n$ LF $(n < m)$. This is because Stage-II of our network estimates convolutional weights for different SAIs during inference and does not freeze the weights at the end of training. 3) Instead of restoring LF views in steps, we restore multiple views in a single forward pass for faster inference. 4) Compared to state-of-the-art, our model achieves 1 dB higher restoration PSNR, with $9 \times$ speedup, 23% smaller model size and at least $5 \times$ lower floating-point operations, as shown in Fig. 1.

2. Related work

LF based methods: Numerous methods have been proposed for LF images addressing various concerns such as increasing the spatial resolution [21, 54, 53, 59, 55], denoising [35, 2, 8, 44, 9], depth estimation [20, 52, 48] and saliency detection [51, 56]. But all these methods mainly consider good lighting conditions. However, under extreme low-light conditions, the signal is heavily corrupted by photon noise with almost no color information. Single-frame
denoising methods such as SGN [13] and BM3D [7] can be used to denoise LF views individually but this does not guarantee epipoles preservation. Consequently, few methods have been proposed for LF denoising. Mitra and Veeraraghavan [35] using the disparity cues modeled 4D LF patches using Gaussian Mixture Models (GMM) and provided a combined algorithm for LF super-resolution and denoising. Dansereau et al. [8] used frequency domain passband filtering for LF denoising and LFBM5D [2] extended the single-frame denoising BM3D algorithm for 4D LF images. Nevertheless, these methods cannot address the color restoration aspect of low light restoration. Secondly, the noise level found in extreme low-light is much greater than what these methods were designed. To address these concerns, recently, L3Fnet [26] and MTO [58] were proposed. L3Fnet used a two-stage deep-learning architecture for restoring low light LFs. In stage-I, L3Fnet extracted the overall 4D LF geometry and in stage-II this information was used to restore each LF view. L3Fnet also released a publicly available low-light LF dataset for training and benchmarking. MTO was, however, only tested for synthetic low-light images.

**Single-frame low-light methods:** Single-frame methods have witnessed a lot of progress towards low-light enhancement. The earliest approaches relied on modifying the image’s histogram [23, 40, 47, 6, 19, 29] to increase its dynamic range. Later approaches, however, used the retinex theory [27, 28] to decompose the low-light image into illumination and reflectance components and use them for low-light enhancement [50, 10, 14, 11, 31, 4, 57, 49]. All these methods have considered weakly illuminated scenes and not night-time extreme low-light conditions. SID [3], a landmark paper on low-light enhancement, used a U-net architecture for extreme low-light single-frame restoration. Since then, several other works have come up aiming to address very low-light conditions [13, 33, 25] but we still find SID having the best tradeoff between speed and accuracy for single-frame images.

### 3. Fast restoration of low-light LF

Fig. 2 shows our three-stage network for restoring \( n \times n \) light field images captured in extreme low-light conditions. The four main challenges addressed by this network towards low-light LF restoration are denoising, epipoles preservation, color restoration and fast inference.

Stage-I looks at all SAIs to have a global understanding. Stage-II operates on specific views and Stage-III exclusively focuses on the local neighborhood of SAIs to be restored. Our RNN inspired feedforward network then fuses these complementary information to restore multiple SAIs in a single forward pass.

#### 3.1. Network architecture

**Stage-I:** Generally, multiple image denoising [32, 15] gives better results than single image denoising because of utilizing the complementary information present in different shots. For a LF camera, however, this complementary information is readily available in a single camera exposure, in the form of SAIs. Thus, in Stage-I, we depth-wise concatenate all the low-light LF views and pass them through a convolutional layer to produce a global activation having the same spatial resolution as LF views. This global activation is necessary to preserve the LF geometry across all LF views.

**Stage-II:** While Stage-I looks at all the SAIs simultaneously, Stage-II operates on SAIs individually. Stage-II randomly selects a non-Peripheral LF view and depth-wise concatenates the global activation to it. The two are then fused using a single convolutional layer. The convolutional layer’s weights are estimated using a series of fully connected layers and angular indices, \((s, t)\), of the chosen SAI.

**Stage-III:** Stage-III selects the \( 3 \times 3 \) neighborhood of the LF view chosen in Stage-II. Stage-I’s global activation, Stage-II’s view-specific output and these nine LF views are then depth-wise concatenated. Our feedforward network then processes this combined tensor to restore the entire \( 3 \times 3 \) neighborhood jointly. Concatenating the \( 3 \times 3 \) neighborhood at the beginning of Stage-III is very crucial. Without this step, it becomes extremely difficult for the network to capture the small baseline between LF views, and as a result, the epipolar geometry of LFs is destroyed in the restored LFs.

For each non-Peripheral view selected in Stage-II, the entire \( 3 \times 3 \) neighborhood gets restored at the end of Stage-III. Thus, to save computation, views with a non-overlapping neighborhood should be selected in Stage-II. But, in cases where \( n \) is not a multiple of 3, there will be instances where a view will be restored twice. In such cases, we randomly chose one of them. More complicated measures such as using a weighted average to combine them may be used, but we did not find them to improve the restoration quality.

**Pre-processing:** Under extreme low-light conditions, the colors captured by any optical system are very poor with low-intensity values. Restoring colors, thus, generally requires amplifying the input image, as adopted by SID [3] and L3Fnet [26]. Much of this amplification in our case is implicitly done by the network by using larger weights and biases for convolutional layers. If, however, the low-light image amplification is also externally supervised to help the network adjust to different lighting conditions, the restoration quality enhances. Thus, taking inspiration from L3Fnet, we compute a six bin histogram from the green channel of the incoming low-light LF and jointly learn the weights for each bin. Using these six weights, we compute a weighted average of the histogram values and multiply it.
parameters, where \( C_i \) and \( C_o \) are the number of input and output channels and \( k \times k \) is the size of the kernel. The global activation of Stage-I and LF SAIs, both have three channels. Thus depth-wise concatenation gives \( C_i = 6 \). \( C_o \) and \( k \) are set to 3.

To estimate convolutional layer’s parameters, we use \( C_i \times C_o \) number of fully connected layers, each having \( k^2 \) nodes. If convolution biases are also to be estimated, as indicated by the second addend in Eq.(1), an additional fully connected layer can be used in the end having \( C_o \) nodes (not shown in Fig. 2). ReLU nonlinearity is present after each layer. Input to these fully connected layers are the angular indices \((s, t)\), and the values obtained at each node, after the forward pass, are reshaped to become the convolutional layer’s parameters. For measuring \((s, t)\), the central SAI is considered the origin. For example, in a \( 9 \times 9 \) LF, the extreme top-left SAI will have \( s = t = -4 \). In summary, if \( f(\cdot) \) denotes the action of fully connected layers, \( \text{cat}(\cdot, \cdot) \) denotes depth-wise concatenation, \( * \) denotes 2D convolution and \( I_{s,t} \) denotes \((s, t)\) SAI of low-light LF, then the output of Stage-II is,

\[
\text{ConvWeights} = \text{reshape} \left[ f(s, t) \right] \\
\text{StageIIo/p} = \text{ConvWeights} \ast \text{cat}(\text{StageIIo/p}, I_{s,t}) \quad (2)
\]

### 3.3. RNN inspired feedforward network

U-net’s multi-scale architecture has been immensely successful in the Computer Vision community but was not designed to model long-range dependencies between different LF views. In contrast, RNNs were specially designed for sequence modeling. We thus analyze a \( N^{th} \) order RNN [46], unfold it in time and propose a set of rules to transform them into a feedforward network.

We model the hidden state \( h^t \) of a \( N^{th} \) order RNN that keeps track of \( N \) preceding states as

\[
h^t = w_0^t \left( w_2^t x^t + \sum_{j=1}^{N} w_j^t \cdot h^{t-j} \right).
\]

Here, \( h^t \) and \( x^t \) are the hidden state and the inputs to the \( N^{th} \) order RNN at timestamp \( t \). The terms denoted by \( w \) are the RNN weights. A larger \( N \) implies more memory units and so to keep the model complexity low, we fix \( N = 2 \) as

with the whole LF before feeding to our network.

### Loss function:

Let \( I \) and \( \hat{I} \) denote the GT and restored LF, \( I_{s,t} \) and \( \hat{I}_{s,t} \) denote SAIs of GT and restored LF, \( F(\cdot) \) denote the amplitude of a 2D DFT operation, \( \psi \) denote the \text{relu}2.2 and \text{relu}3.3 convolutional layers [22] of VGG-16 architecture. Then the loss function for training is,

\[
\frac{0.2}{N^2} \sum_{s,t} \left( ||F(I_{s,t}) - F(\hat{I}_{s,t})||_1 + ||\psi(I_{s,t}) - \psi(\hat{I}_{s,t})||_1 \right) + ||I - \hat{I}||_1
\]

Occasionally the L3F dataset [26] exhibits very small translational misalignment. Thus, we use DFT amplitude loss, which is invariant to small translational shifts in signal. We could have also used contextual loss [34], but this slows down the training by a factor of \( 3 \times \) with almost no gain in performance.

### 3.2. Estimating weights in Stage-II

Stage-I and Stage-III of our network share weights across all SAIs. ResLF [59] instead used a different set of network weights for each SAI for superior restoration but at the expense of significantly increasing the model size. Thus, in our network only Stage-II uses view-specific convolutional weights. One drawback with this approach is that this limits the model in restoring only those SAIs during inference whose weights were learned during training. We, however, want our network to be flexible enough to restore a \( 9 \times 9 \) LF even if trained for a smaller LF, say \( 7 \times 7 \) LF. Thus, we resort to learning a mechanism during training that can estimate the convolutional weights using SAIs angular location and whose parameters do not depend on the number of views to be restored.

A convolutional layer requires,

\[
\left( k^2 \times C_i \times C_o \right) + C_o \\
\quad \text{(1)}
\]

parameters, where \( C_i \) and \( C_o \) are the number of input and output channels and \( k \times k \) is the size of the kernel. Functionally identical to Stage-III’s feedforward network in Fig. 2.

Figure 3. Various steps involved in designing the feedforward network used in Stage-III of the proposed solution.
We replace all element-wise addition operations with channel-wise concatenation operation.

2. At each timestamp, the input \( x^t \) is the channel-wise concatenated global activation of Stage-I, view-specific output of Stage-II and \( 3 \times 3 \) neighborhood from Stage-III and is denoted by \( x \).

3. RNN weights which lead to the formation of a feature map are Residual blocks \([17]\) with \( 3 \times 3 \) kernel. In our case these weights are \( w_0 \) (denoted in red in Fig. 3 b).

As after unrolling a RNN into a feedforward network, the conception of timestamp \( t \) becomes meaningless, \( w_0 \) will be denoted simply as \( w_i \), where \( i \) is an integer.

4. All other weights are either up/down sampling operations, implemented using Pixel-Shufle [45, 13].

5. If a weight is looking at \( N^{th} \) preceding state, then it can perform up/down sampling operation by a factor of \( 2^\alpha \), where \( \alpha \in \{0, N\} \).

Based on the above set of suggested rules, several feedforward networks are possible depending on how many timestamps the RNN is unfolded. Fig. 3 c) and d) show some feedforward networks obtained using the above stated rules, where \( w_i \)'s are Residual blocks with \( 3 \times 3 \) kernels. Infact, Fig. 3 c) is functionally identical to Stage-III’s RNN inspired feedforward network shown in Fig. 2. The exact mapping of weights from unfolded \( 2^{nd} \) order RNN, shown in Fig. 3 c), to our Stage-III feedforward network is given in Tab. 1. More discussion can be found in supplementary. Key features of these feedforward networks are:

- Each Residual block has direct access to the input \( x \) consisting of \( 3 \times 3 \) neighborhood of the SAI, leading to better epipoles preservation.
- As we had started with \( N = 2 \) order RNN, each Residual Block can directly access \( N \) preceding feature maps. Thus, a larger \( N \) offers much greater expressiveness but with proportionally larger time and memory complexity that might be unnecessary.
- The number of scale-spaces we can have depends on how many timestamps \( t \) we unfold the RNN.

## 4. Experiments

### 4.1. Experimental settings

We used PyTorch [39] running on Intel Xeon E5-1620V4 CPU with 64GB RAM and GTX 1080Ti GPU. The network shown in Fig. 2 was initialized using MSRA [16] initialization and trained for 50,000 iterations using ADAM optimizer [24] with a learning rate of \( 10^{-4} \). Weight normalization [43] was used for each convolutional layer. All three stages were trained end-to-end for about 10 hours on a single GPU. During training \( 128 \times 128 \) patches were randomly cropped from each SAI with batch size of 2 and employing horizontal and vertical flipping. For testing, we used full spatial resolution.

We used the publicly available Low-Light-Light-Field (L3F) dataset [26] collected using commercially available Lytro Illum for benchmarking the proposed solution. L3F dataset was collected in the evening when the light intensity falling on the camera lens was on an average of about 10 lux. Ground truth (GT) images were captured using longer exposure time ranging from 1 – 30 seconds. The optimal GT exposure time was then reduced by \( 20\times, 50\times \) and \( 100\times \) to capture the low-light LFs and were arranged into L3F-20, L3F-50 and L3F-100 subsets. The L3F-100 is the most challenging amongst the three subsets. Each set consists of 27 scenes, of which 9 are reserved for testing. Each LF consists of \( 15 \times 15 \) views. Peripheral views suffer ghosting and vignetting artifacts \([59, 53]\), and the prevailing practice is to ignore views equally from all boundaries \([59, 53, 54, 55, 35]\) and evaluate for central \( n \times n \) views, \( n \in [5, 7, 9] \). But oddly, L3Fnet ignored more views from the top and left boundary and evaluated for central \( 8 \times 8 \) views. To align our evaluation with most existing works on LFs, we evaluate all algorithms for central \( 9 \times 9 \) views.

We also show results on the Stanford General Light Field dataset [41] by simulating low-light conditions. To simulate low-light, we first divide the intensity by \( s \in [9, 11] \) and then darken it using gamma correction with \( \gamma \in [1.5, 2] \). Finally, signal-dependent Poisson noise is added to simulate the photon noise. The dataset consists of 57 scenes, of which 17 were reserved for testing.

We compare the proposed method against 9 existing methods, namely PBS [57], RetinexNet [4], DID [33], SGN [13], SID [3], LFBM5D [2], MTO [58], ResLF [59]...
indicates second-best.

To adapt ResLF for low-light restoration we removed the last Pixel-Shuffle block to match input and output spatial resolution and retrained it on L3F dataset. We also observed that ResLF only operates on intensity channel and simply extrapolates the color channels. This may be alright for recovering high frequency details for Super-Resolution tasks but not for color enhancement and denoising operation required for low-light restoration. We thus re-trained the network with all RGB channels. We also tried using our loss function instead of just L1 loss used by ResLF. All above modifications to ResLF gave atleast 3dB higher PSNR, and so we use this version for comparisons. Likewise, MTO was majorly designed for grayscale synthetic low-light LF and so naturally its performance on real low-light RGB LF was poor. So we re-trained it with RGB low-light LFs and use this version for comparisons. Finally, a limitation of L3Fnet is that it cannot restore peripheral views. Thus to get 9 × 9 restored LF from L3Fnet it was given additional information and was provided with 11 × 11 central SAIs. The code will be available at mohitlamba94.github.io/DarkLightFieldRestoration

### 4.2. Comparison with existing methods

**Quantitative comparisons.** Tab. 2 presents a quantitative comparison of our method with existing approaches on the real low-light L3F dataset. Our method significantly outperforms all existing approaches in terms of PSNR/SSIM metrics. The L3F-100 dataset is very challenging compared to the L3F-20 and L3F-50 datasets because of extremely low pixel intensity and significant photon noise. Thus all methods have the lowest PSNR/SSIM
for the L3F-100 dataset and the highest for the L3F-20 dataset. A side evidence of the effectiveness of our method is that the difference in the restoration quality between the L3F-20 and L3F-100 dataset is only about 2dB for us, but for other methods such as PBS it is more than 7dB.

**Qualitative comparisons.** Methods like PBS, RetinexNet and LFBM5D have quite noisy restorations. These methods are appropriate for mild denoising but cannot tackle excessive photon noise in very dark conditions. On the contrary, restorations of methods like SID and SGN are quite blurry as they do not jointly utilize information from all SAlrs. Though we re-trained MTO on real low-light LFs, it struggles to recover colors from low-light LFs. Most results shown in MTO paper [58] are for grayscale synthetic low-light LFs. Although, ResLF uses a much deeper network, its restorations are lower than ours. This is mostly because, unlike U-net architecture, ResLF does not perform multi-scale processing and so has a very small receptive field (about 25 x 25). This may be sufficient for recovering details for super-resolution task but not for color restoration and noise suppression in real low-light LFs. Compared to the current state-of-the-art L3Fnet, our restorations better preserve finer details. This fact is also corroborated by additional visual comparisons shown Fig. 5. Quantitatively also, L3Fnet achieves a PSNR/SSIM of 28.76dB/0.85 on the Stanford general light field dataset, while our method achieves 29.30dB/0.86.

**Computational Complexity.** Tab. 3 shows the computational complexity of the top-performing models for restoring a 9 x 9 LF with 432 x 624 spatial resolution. We observe that our method offers a significant improvement for every metric. Specifically, compared to state-of-the-art L3Fnet, we have a 23% smaller model size (i.e number of parameters), use about 5 x lower floating point operations (i.e GMACS) and about 9 x faster on both GPU and CPU. The main reason for our extremely fast inference is that we requires only 9 forward pass of Stage-II and Stage-III to re-store a single 9 x 9 LF. In contrast, other methods require 81 forward pass. One final limitation of L3Fnet is that, given a n x n LF, it can restore only central n - 2 x n - 2 views.
In doing so we did not change the number of model parameters nor the number of convolutional layers. The PSNR dropped from 23.45 dB to 23.02 dB. This is expected because in contrast to U-net, each residual block present in our RNN inspired feedforward has direct access to input LF views and up to 2 preceding feature maps.

In the second ablation study, we use only a single convolutional layer in Stage-II and share its weights across all SAIs. Consequently, the network now becomes oblivious to the angular location of the SAIs and we find the PSNR drops to 23.10 dB.

In the third ablation study, we do not feed the 3×3 neighborhood to our RNN inspired feedforward network. The PSNR significantly drops to 22.58 dB and we also observed that the epipoles were not appropriately restored. Besides Stage-II and Stage-III, Stage-I is required to preserve epipolar geometry across views, as demonstrated in L3Fnet.

In the fourth ablation study, we replaced our feedforward network present in Stage-III with the one unrolled for t = 5 instead of t = 3. This increased the PSNR to 23.69 dB but with much greater computational complexity. We thus continue to use the t = 3 feedforward network.

Finally, re-training the network without the DFT loss causes significant drop in PSNR.

5. Conclusion

We proposed a novel three-stage network that can be trained end-to-end to utilize three complementary information present in real low-light LFs, namely global, local and view-specific. These complementary information were then fused together by our RNN inspired feedforward network to restore very low-light LFs. Additionally, the network restores multiple SAIs in a single forward pass for significantly faster inference. Overall, the combined effect of these contributions were that compared to state-of-the-art L3Fnet, we achieved up to 1 dB higher restoration PSNR, with 9× speedup, 23% smaller model size and about 5× lower floating-point operations. Finally, we also showed that our network is flexible enough to restore a 9 × 9 LF during inference even if trained for 5 × 5 or 7 × 7 LF.

<table>
<thead>
<tr>
<th>PSNR (dB)</th>
<th>SSIM</th>
<th>SID</th>
<th>MTO</th>
<th>ResLF</th>
<th>L3Fnet</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.01</td>
<td>0.48</td>
<td>19.31</td>
<td>0.51</td>
<td>21.10</td>
<td>0.56</td>
<td>21.90</td>
</tr>
</tbody>
</table>

Table 4. Average PSNR/SSIM of EPIs constructed from LFs restored by different methods on L3F-100 dataset.

| Use U-net in Stage-III | 23.02/0.70 |
| Share conv weights in Stage-II | 23.10/0.70 |
| No neighbouring SAIs in Stage-II | 22.58/0.68 |
| Unfold RNN for t = 5 instead of t = 3 | 23.69/0.71 |
| Training without DFT loss | 22.17/0.68 |
| Proposed | 23.45/0.71 |

Table 5. Ablation studies for our network on the L3F-100 dataset. Though using the feedforward network unrolled for t = 5 has higher PSNR, it has much greater time and computational complexity and we prefer using the feedforward network with t = 3.
References


