Deep Guided Learning for Fast Multi-Exposure Image Fusion

Kede Ma, Member, IEEE, Zhengfang Duamu, Student Member, IEEE, Hanwei Zhu, Student Member, IEEE, Yuming Fang, Senior Member, IEEE, and Zhou Wang, Fellow, IEEE

Abstract—We propose a fast multi-exposure image fusion (MEF) method, namely MEF-Net, for static image sequences of arbitrary spatial resolution and exposure number. We feed a low-resolution version of the input sequence to a fully convolutional network for weight map prediction. We then jointly upsample the weight maps using a guided filter. The final image is computed by a weighted fusion. Unlike conventional MEF methods, MEF-Net is trained end-to-end by optimizing the perceptually motivated MEF structural similarity (MEF-SSIM) index over a database of training sequences at full resolution. Across an independent set of test sequences, we find that the optimized MEF-Net achieves consistent improvement in visual quality for most fused images and runs 10 to 1000 times faster than state-of-the-art methods. The code will be made publicly available at https://github.com/makedede/MEFNet.

Index Terms—Multi-exposure image fusion, convolutional neural networks, guided filtering, computational photography.

I. INTRODUCTION

MULTI-EXPOSURE image fusion (MEF) provides a cost-effective solution for high-dynamic-range (HDR) imaging [1]. It takes an image sequence with different exposure levels as input and produces a high-quality and low-dynamic-range image, ready for display [2]. Research in MEF has yielded a number of methods [2]–[8], which generate fused images with faithful detail preservation and vivid color appearance. This is mainly accomplished by a weighted summation framework

\[ Y = \sum_{k=1}^{K} W_k \odot X_k, \]  

where \( \odot \) denotes the Hadamard product; \( W_k \) and \( X_k \) represent the \( k \)-th weight map and the corresponding exposure image, respectively; \( Y \) is the fused image; and \( K \) is the number of exposures in the input sequence. Noticeable exceptions of the framework are optimization-based methods [6], [8], where the fusion process is supervised by a perceptual image quality metric [9].

Despite the demonstrated success, the high resolution of the exposure sequence captured by commercial cameras and mobile devices poses a grand challenge to existing MEF methods, which may require extensive computational resources and take seconds (or even minutes) to generate the fused image. The situation becomes even worse with the increasing number of exposures. Algorithm acceleration through code optimization is possible [10], [11], but it may not generalize across different MEF methods. Another general approach to accelerate an MEF method [12]–[14] is to downsample the input sequence, execute the MEF operator at low resolution, and upsample the fused image. One drawback of this approach is that the MEF method never sees the high-resolution sequence and therefore fails to fully reproduce its fine details, limiting the visual sharpness of the fused image.

We aim to develop an MEF method for static scenes with three desirable properties:

- **Flexibility.** It must accept input sequences of arbitrary spatial resolution and exposure number.
- **Speed.** It must be fast, facilitating real-time mobile applications at high resolution.
- **Quality.** It must produce high-quality fused images across a broad range of content and luminance variations.

To achieve flexibility, we utilize a fully convolutional network [15], which takes an input of arbitrary size and produces an output of the corresponding size (known as dense prediction). The network is shared by different exposed images, enabling it to process an arbitrary number of exposures. To achieve speed, we follow the downsample-execute-upsample scheme and feed the network a low-resolution version of the input sequence. Rather than producing the fused image as in [6], [16], [17], the network learns to generate the low-resolution weight maps in Eq. (1) and jointly upsample them using a guided filter [18] for final weighted fusion. By doing so, we take advantage of the smooth nature of the weight maps and make use of the input sequence as the guidance [19]. Directly upsampling the fused image is difficult due to the existence of rich high-frequency information in the high-resolution sequence and the lack of proper guidance. To achieve quality, we integrate the differentiable guided filter with the preceding network [19] and optimize the entire model end-to-end for the subject-calibrated MEF structural similarity (MEF-SSIM) index [9] over a large number of training sequences [6], [20]–[23]. Although most of our inference and learning is performed at low resolution, the objective function MEF-SSIM [9] is measured at full resolution, which encourages the guided filter to cooperate with the convolutional network, generating high-quality fused images. Extensive experiments demonstrate that the resulting MEF-Net achieves consistent improvement in
visual quality compared with state-of-the-art MEF methods for most sequences. More importantly, MEF-Net runs 10 to 1000 times faster and holds much promise for approximating and accelerating the MEF methods that are computationally intensive.

II. RELATED WORK

In this section, we provide a brief overview of existing MEF methods and general approaches for fast image processing, with emphasis on previous ones that are closely related to our work.

A. Existing MEF Algorithms

The Laplacian pyramid [24] proposed by Burt and Adelson in 1983 has a lasting impact on image fusion research [25]. Combining with Gaussian [2], [3] or edge-preserving [4], [7] filters, the Laplacian pyramid provides a convenient multi-resolution framework to refine the weight map $W_k$, which carries perceptually important information of $X_k$. Mertens et al. [2] adopted this framework and proposed one of the first pixel-wise MEF methods, which keeps a good balance between visual quality and computational complexity. Since then, a great number of pixel-wise MEF methods [26] have been developed, mainly to improve visual quality at the cost of increasing computational complexity. Compared to pixel-wise MEF, patch-wise methods generally produce a smoother $W_k$ that requires less post-processing, but bear heavier computational burdens. Goshtasby [27] presented one of the first patch-wise MEF methods. Ma and Wang [28] extended the idea [27] and developed a structural patch decomposition for MEF. Typical perceptual factors that contribute to $W_k$ include gradient [29], contrast [2], color saturation [2], [7], entropy [27], structure [28], well-exposedness [2], [3], and saliency [4].

B. Fast Image Processing

As mobile devices become people’s primary cameras to take photos, there is a growing demand to accelerate image processing operators for novel mobile applications such as photo editing, face manipulation, and augmented reality. A good case in point is bilateral filtering [30]–[32], which benefits from years of code optimization, due to the ubiquity of edge-preserving image processing. However, such acceleration tricks may not generalize to other operators. A system-level acceleration solution, friendly to mobile hardware, is to send images to a cloud server, execute the image processing operator on the cloud, and send the processed images back [33]. Due to the large bitrate of high-resolution images, this may introduce significant delays, especially when the network condition is unstable. The downsample-execute-upsample scheme is another general method for algorithm acceleration, which suffers from two limitations. First, the underlying operator may still be slow to run at low resolution, which may still be slow. Second, it is difficult for upsampling techniques to recover the high-frequency information in the high-resolution images, especially when they are of complex structures. Recently, due to their efficient feed-forward inference, convolutional networks [14], [19] have been used to approximate and accelerate popular image processing operators in edge-preserving filtering, detail manipulation, non-local dehazing, and style transfer.

C. Closely Related Work

Our work is closely related to several previous methods. Li et al. [4] first introduced guided filtering into MEF. The weight map $W_k$ was constructed based on pixel saliency and spatial consistency measurements, and was refined by a guided filter. Kou et al. [7] built their work upon [2] and replaced Gaussian smoothing with gradient domain guided filtering. The three components of the above two methods—weight map construction, guided filtering, and weighted fusion—are optimized separately (often through manual adjustment). Our method differentiate from them by resorting to an end-to-end solution, where the three components are jointly optimized in a data-driven fashion. Rather than pre-defining a computational graph for MEF, Ma et al. [8] formulated it as an optimization problem

$$Y_{opt} = \arg \max_Y \text{MEF-SSIM} \{\{X_k\}, Y\}$$
subject to $0 \leq Y \leq 255$.

Due to the nonconvexity of MEF-SSIM [9] and the high-dimensionality of the optimization problem, a closed-form solution is difficult. Therefore, a gradient-based iterative solver is adopted [8], which is computationally expensive. Another work closely related to ours is from Prabhakar et al. [6], who trained a feed-forward convolutional network to solve the optimization problem in (2). The method works reasonably well on extreme situations, but does not achieve the flexibility, speed, and quality we seek. We will show that the proposed MEF-Net achieves higher quality, while being much faster and more flexible.

Chen et al. [14] investigated a number of convolutional network architectures in terms of their approximation accuracy, speed, and compactness when accelerating image processing operators. They found that a multi-scale context aggregation network (CAN) characterized by dilated convolutions [34] satisfies the three criteria and significantly outperforms prior methods [13]. We adopt CAN as our default network architecture. Wu et al. [19] treated the guided filter as a group of spatially-varying differentiable transformations and integrated it with convolutional networks for end-to-end training. Although their method [19] achieves superior performance in some applications with relatively smooth outputs (e.g., style transfer [35]), it cannot accurately approximate operators that work with high-frequency image content (e.g., multi-scale tone manipulation [36]). Our method avoids this problem by applying the guided filter on $W_k$, which is smooth in nature and easy to upsample.

III. MEF-Net

We describe MEF-Net, a flexible, fast, and high-quality MEF method. MEF-Net consists of a bilinear downsampler,
A. CAN for Low-Resolution Weight Map Prediction

The core module of MEF-Net is a convolutional network, which transforms the low-resolution input sequence \( \{ X_k \} \) to predict the low-resolution weight maps \( \{ W_k \} \). Taking \( \{ W_k \}, \{ X_k \}, \) and \( \{ X_k \} \) as inputs, we obtain the high-resolution weight maps \( \{ W_k \} \) using the guided filter, an operation known as joint upsampling in computer vision [37]. Finally, we compute the fused image \( Y \) using Eq. (1). MEF-Net is end-to-end trainable with the objective function MEF-SSIM [9] evaluated at high resolution.

The architecture is shown in Fig. 1. We first downsample an input sequence \( \{ X_k \} \) and feed the CAN the low-resolution version \( \{ X_k \} \) to predict the low-resolution weight maps \( \{ W_k \} \). Taking \( \{ W_k \}, \{ X_k \}, \) and \( \{ X_k \} \) as inputs, we obtain the high-resolution weight maps \( \{ W_k \} \) using the guided filter, an operation known as joint upsampling in computer vision [37]. Finally, we compute the fused image \( Y \) using Eq. (1). MEF-Net is end-to-end trainable with the objective function MEF-SSIM [9] evaluated at high resolution.

A. CAN for Low-Resolution Weight Map Prediction

The core module of MEF-Net is a convolutional network, which transforms the low-resolution input sequence \( \{ X_k \} \) to predict the low-resolution weight maps \( \{ W_k \} \). Taking \( \{ W_k \}, \{ X_k \}, \) and \( \{ X_k \} \) as inputs, we obtain the high-resolution weight maps \( \{ W_k \} \) using the guided filter, an operation known as joint upsampling in computer vision [37]. Finally, we compute the fused image \( Y \) using Eq. (1). MEF-Net is end-to-end trainable with the objective function MEF-SSIM [9] evaluated at high resolution.

B. Guided Filter for High-Resolution Weight Map Upsampling

The key assumption of the guided filter is a local linear model between the guidance \( I \) and the filtering output \( Q \) [18]

\[
Q(i) = a_i I(i) + b_\omega, \forall i \in \omega,
\]

where \( i \) is the index of the guidance and \( \omega \) is a local square window with radius \( r \). \( a_\omega, b_\omega \) are linear coefficients assumed
Algorithm 1: Guided filtering for joint upsampling in MEF-Net

**Input:** low-resolution weight map \( W_k \), low-resolution \( X_k \) as guidance, high-resolution \( X_k \) as guidance, radius \( r \), and regularization \( \lambda_a \)

**Output:** high-resolution weight map \( W_k \)

1: mean\(_g\) = \( f_{mean} (X_k^i) \)
   mean\(_i\) = \( f_{mean} (W_k^i) \)
   corr\(_g\) = \( f_{mean} (X_k^i \odot X_k^i) \)
   corr\(_{gi}\) = \( f_{mean} (X_k^i \odot W_k^i) \)
   var\(_g\) = corr\(_g\) - mean\(_g\)\odot mean\(_g\)
   cov\(_{gi}\) = corr\(_{gi}\) - mean\(_g\)\odot mean\(_i\)
   \( A_k^i = \text{cov}_{gi} \odot (\text{var} + \lambda_a) \)
   \( B_k^i = \text{mean}_i - A_k^i \odot \text{mean}_g \)
2: \( A_k = f_I (A_k^i) \)
   \( B_k = f_I (B_k^i) \)
3: \( W_k = A_k \odot X_k + B_k \)

Algorithm 1 summarizes the guided filter for joint upsampling in MEF-Net, where \( f_{mean} \) and \( f_I \) denote box filtering and bilinear upsampling, respectively. By interpreting the guided filter as a group of spatially-varying differentiable transformations [19], we integrate it with the preceding CAN and optimize MEF-Net end-to-end at full resolution. We may apply the guided filter as a post-processing step without any training, but it hurts the fusion performance, as will be clear in Section IV-B.

To stabilize gradients during training and to obtain consistent results, we take the absolute values of \( \{W_k\} \) followed by normalization such that they sum to one across exposures at each spatial location \( i \)

\[
\hat{W}_k(i) = \frac{|W_k(i)|}{\sum_{k=1}^{K} |W_k(i)|}.
\]

Fig. 2 demonstrates the learned weight maps \( \{\hat{W}_k\} \) of the source sequence “Corridor”, where a bright pixel in \( W_k \) indicates that the corresponding pixel in \( X_k \) contributes more to the fused image \( Y \). The learned \( W_k \) enjoys several desirable properties. First, \( W_k \) is smooth with gentle transitions from sharp to flat regions. Second, \( W_k \) prefers high-contrast and well-exposed regions, both of which significantly impact the perceptual quality of the fused image. Third, \( W_k \) reflects the global structure of \( X_k \) and is beneficial for large-scale detail preservation. As a result, the fused image appears natural without loss of details and presence of artifacts.

### C. MEF-SSIM as Objective Function

In this subsection, we detail the MEF-SSIM index [9] as the objective function for MEF-Net. Other perceptual quality metrics for MEF such as [41], [42] may also serve the purpose. Specifically, MEF-SSIM decomposes an image patch \( x_k \) into three conceptually independent components

\[
x_k = \| x_k - \mu_{x_k} \| \frac{x_k - \mu_{x_k}}{\| x_k - \mu_{x_k} \|} + \mu_{x_k} = \| \tilde{x}_k \| \cdot \frac{\tilde{x}_k}{\| \tilde{x}_k \|} + \mu_{x_k} = c_k \cdot s_k + l_k,
\]

where \( \| \cdot \| \) denotes the \( l_2 \)-norm. \( l_k = \mu_{s_k} \), \( c_k = \| \tilde{x}_k \| \), and \( s_k = \tilde{x}_k / \| \tilde{x}_k \| \) represent the intensity, contrast, and structure of \( x_k \), respectively [9].

The desired intensity of the fused image patch is defined by

\[
\hat{l} = \frac{\sum_{k=1}^{K} w_l (\mu_{k}, l_k) l_k}{\sum_{k=1}^{K} w_l (\mu_{k}, l_k)},
\]
where \( w_l(\cdot) \) is a weight function of the global mean intensity \( \mu_k \) of \( X_k \) and the local mean intensity \( l_k \) of \( x_k \). \( w_l(\cdot) \) is specified by a two-dimensional Gaussian profile

\[
\hat{w}_l(\mu_k, l_k) = \exp \left( -\frac{(\mu_k - \tau)^2}{2\sigma_0^2} - \frac{(l_k - \tau)^2}{2\sigma_1^2} \right), \tag{11}
\]

where \( \sigma_0 \) and \( \sigma_1 \) are two photometric spreads, set to 0.2 and 0.5, respectively \([5]\). \( \tau = 128 \) represents the mid-intensity value for an 8-bit sequence. The desired contrast is determined by the highest contrast in \( \{x_k\} \)

\[
\hat{c} = \max_{1 \leq k \leq K} c_k. \tag{12}
\]

The desired structure is computed by a weighted summation

\[
\hat{s} = \frac{s}{\|s\|}, \quad \text{where} \quad s = \frac{\sum_{k=1}^{K} w_s(\hat{x}_k) s_k}{\sum_{k=1}^{K} w_s(\hat{x}_k)}, \tag{13}
\]

where \( w_s(\cdot) = \|\cdot\|_\infty \) is an \( \ell_\infty \)-norm weight function.

Once \( \hat{l}, \hat{c}, \) and \( \hat{s} \) are determined, we invert the decomposition to obtain the desired fused patch

\[
\hat{x} = \hat{c} \cdot \hat{s} + \hat{l}. \tag{14}
\]

The construction of MEF-SSIM follows the definition of the SSIM index \([43]\)

\[
S(\{x_k\}, y) = \frac{2\mu_{\hat{x}}\mu_y + C_1}{(\mu_{\hat{x}}^2 + \mu_y^2 + C_2)} \frac{2\sigma_{\hat{x}y} + C_2}{\sigma_{\hat{x}}^2 + \sigma_y^2 + C_2}, \tag{15}
\]

where \( \mu_{\hat{x}} \) and \( \mu_y \) denote the mean intensities of the desired patch and a given fused patch, respectively. \( \sigma_{\hat{x}}, \sigma_y, \) and \( \sigma_{\hat{x}y} \) denote the local variances of \( \hat{x} \) and \( y \), and their covariance, respectively. \( C_1 \) and \( C_2 \) are two small positive constants to prevent instability. The local \( S \) values are averaged to obtain an overall quality measure of the fused image

\[
\text{MEF-SSIM}(\{X_k\}, Y) = \frac{1}{N} \sum_{i=1}^{N} S(\{R_i X_k\}, R_i Y), \tag{16}
\]

where \( R_i \) is a matrix that extracts the \( i \)-th patch from the image. The MEF-SSIM score ranges from 0 to 1 with a higher value indicating better visual quality.

The vanilla version of MEF-SSIM \([9]\) excludes the intensity comparison and has been adopted by Prabhakar et al. \([6]\) to drive the learning of convolutional networks for MEF. In our experiments, we find that optimizing MEF-SSIM without intensity information is unstable, resulting in fused images with a relatively pale appearance (see Fig. 6). The improved version of MEF-SSIM \([8]\) adds the intensity comparison in Eq. (15) and directly works with color sequences. Nevertheless, it is likely to generate over-saturated colors in some cases \([8]\). To obtain more consistent and conservative fused images with little artifacts, we choose to handle chroma components separately as suggested in \([6]\). Specifically, we work with the \( Y'CbCr \) format and evaluate MEF-SSIM only on the luma components of \( \{X_k\} \) and \( Y \). In other words, the CAN in MEF-Net is optimized to fuse the luma components. For the \( Cb \) chroma components, we adopt a simple weighted summation suggested in \([6]\)

\[
\hat{b} = \frac{\sum_{k=1}^{K} w_c(b_k)b_k}{\sum_{k=1}^{K} w_c(b_k)}, \tag{17}
\]

where \( b_k \) denotes the \( Cb \) chroma value at the \( k \)-th exposure and \( w_c(b_k) = \|b_k - \tau\|_1 \) is an \( \ell_1 \)-norm weight function. The \( Cr \) chroma components can be fused in the same way. Finally, we convert the fused image from \( Y'CbCr \) back to RGB.
Finally, we evaluate MEF-Net at full resolution during testing.

than train MEF-Net on sequences of varying high resolutions larger and state-of-the-art MEF methods in terms of visual quality sequences to MEF-SSIM are inherited from [8], [9]. We resize the exposure only in order to reduce GPU memory cost. The parameters of 600 between three and nine. We train MEF-Net on the maximum epoch number of exposures in the current sequence. The learning stops when Adam are set by default. The batch size is equal to the number solver [45] with a learning rate of 10−4, λ is a critical parameter in MEF-Net, as will be clear in Section IV-B. Training uses the Adam regularizer parameter λr of LReLU is fixed to 0.2. The radius r and the regularization parameter λs of the guided filter are set to 1 and 10−4, respectively. λa is a critical parameter in MEF-Net, as will be clear in Section IV-B.

D. Training

We collect a large-scale dataset for MEF-Net. Initially, we gather more than 1,000 exposure sequences mainly from the five sources [6], [20]–[23] with permission. We first eliminate sequences that contain visible object motion. For camera motion, we retain those sequences that have been successfully aligned by existing image registration algorithms [44]. After screening, a total of 690 static sequences remain, which span a great amount of HDR content, including indoor and outdoor, human and still-life, day and night scenes. Some representative sequences are shown in Fig. 3. The spatial resolution ranges from 0.2 to 20 megapixels, while the number of exposures is between three and nine. We train MEF-Net on 600 sequences and leave the remaining 90 for testing.

During training, we apply MEF-SSIM on the finest-scale only in order to reduce GPU memory cost. The parameters of MEF-SSIM are inherited from [8], [9]. We resize the exposure sequences to 128s and 512s as the low- and high-resolution inputs to MEF-Net, respectively, where 128s means that the short size is resized to 128 while keeping the aspect ratio. The leaky parameter λr of LReLU is fixed to 0.2. The radius r and the regularization parameter λs of the guided filter are set to 1 and 10−4, respectively. λa is a critical parameter in MEF-Net, as will be clear in Section IV-B. Training uses the Adam solver [45] with a learning rate of 10−4. Other parameters in Adam are set by default. The batch size is equal to the number of exposures in the current sequence. The learning stops when the maximum epoch number 100 is reached. We try to further train MEF-Net on sequences of varying high resolutions larger than 512s [19], but this does not yield noticeable improvement. Finally, we evaluate MEF-Net at full resolution during testing.

A. Main Results

1) Qualitative Comparison: We compare MEF-Net with six previous MEF methods, including Mertens09 [2], Li13 [4], SPD-MEF [5], GGIF [7], DeepFuse [6], and MEF-Opt [8]. Mertens09 [2] is the primary baseline in MEF. Li13 [4] introduces guided filtering [18] into MEF, while GGIF [7] applies guided filtering in the gradient domain and achieves the best performance in a recent subjective experiment [20]. SPD-MEF is an MEF-SSIM-inspired non-iterative method and ranks number two in the same subjective study [20]. MEF-Opt [8] is an gradient-based iterative method, optimizing MEF-SSIM [9] in the space of all images. DeepFuse [6] is a closely related method that trains a convolutional network for MEF. In principle, MEF-Opt can be regarded as an upper bound of all MEF methods in terms of MEF-SSIM. The fused images are generated by the implementations from the original authors with default settings. Since DeepFuse takes two exposures only, we try several under- and over-exposed combinations, and choose the fused image that achieves the best MEF-SSIM for comparison.

Fig. 4 compares Mertens09 [2] and SPD-MEF [5] with MEF-Net on the source sequence “Studio”. As can be seen, Mertens09 does not recover the details of the lamp due to the extreme HDR of the scene and excessive Gaussian smoothing of the weight maps. In addition, the outside ground appears over-exposed. SPD-MEF does a good job in detail and color preservation of the indoor scene, but introduces annoying color and halo artifacts out of the window. We believe the distortions arise because SPD-MEF prefers strong or even over-saturated colors, whose weight maps fail to make smooth transitions across exposures near strong edges. By contrast, MEF-Net

IV. EXPERIMENTS

In this section, we first compare MEF-Net with classic and state-of-the-art MEF methods in terms of visual quality and computational complexity. We then conduct a series of ablation experiments to identify the core components of MEF-Net. Last, we treat MEF-Net as a universal MEF approximator and use it to accelerate existing MEF methods.
produces a more natural appearance with faithful detail and color reproduction.

Fig. 5 compares Li13 [4] and GGIF [7] with MEF-Net on the source sequence “Lake forest”. By decomposing the input sequence into the base and detail layers with Gaussian filtering, Li13 focuses on fine-detail enhancement only and fails to capture large-scale luminance variations. Consequently, apparent halo artifacts emerge. Moreover, the global intensity of the fused image changes abruptly, resulting in an artificial and uncomfortable appearance. Inheriting the multi-scale Laplacian decomposition from Mertens09 [2], GGIF alleviates the halo artifacts to a just noticeable level, but at the same time reduces the global contrast. The fused image looks relatively pale and less detailed. Compared to GGIF, MEF-Net better preserves the global contrast and the overall appearance of the fused image is more natural and appealing.

Fig. 6 compares DeepFuse [6] and MEF-Opt [8] with MEF-Net on the source sequence “Archway”. The fusion performance of DeepFuse depends highly on the quality of the input image pair. If the under- and over-exposed images are not perfectly complementary, DeepFuse may generate a fused image of lower perceptual quality than a normally exposed shot. With only two exposures, it is difficult for DeepFuse to determine the lighting condition of the scene. The missing intensity component of MEF-SSIM during optimization makes the situation worse. As a result, we observe unnatural colors around the two lamps and detail loss on the wall and floor. By operating in the space of all images, MEF-Opt has more freedom than MEF-Net to produce the fused image with finer details, which is supported by a higher MEF-SSIM value. With a sensible network architecture, MEF-Net closely matches the details, which is supported by a higher MEF-SSIM value. With a sensible network architecture, MEF-Net closely matches the

2) Quantitative Comparison: We list the quantitative comparison results in terms of MEF-SSIM [9] in Table II. It is not surprising that MEF-Opt [8] achieves the best performance because it optimizes MEF-SSIM in the space of all images. Among the rest of the methods, MEF-Net is closest to this upper bound, which suggests that the training is highly effective and MEF-Net generalizes well to novel content. Although sharing the same spirit of MEF-SSIM optimization, DeepFuse [6] performs the worst due to the extremely strict constraint on the input sequence. We also employ another subject-calibrated quality model specifically for MEF, namely MEF-VIF [42], to quantify the fusion performance on the same 90 test sequences. From Table II, we see that MEF-Net is among the best performing methods. The proposed MEF-Net is flexible and may be trained to optimize MEF-VIF or other MEF quality measures.

We take a closer look at the cross resolution generalizability of MEF-Net. Specifically, we downsample the 90 test sequences to seven resolutions if possible, ranging from 512s to 2048s, and report the average MEF-SSIM scores in Fig. 7. Despite the fact that MEF-Net is trained on the resolution of 512s, it generalizes remarkably well across a wide range of unseen resolutions with slight MEF-SSIM decrease, which may be explained by the increasing resolution ratio before and after the guided filter. Meanwhile, we observe a steady uptrend of MEF-Opt [8] optimized for MEF-SSIM with the increasing resolution. This may arise because for most MEF algorithms including MEF-Opt, it is easier to fuse flat regions than structured ones; when the spatial resolution increases, the flat regions grow more rapidly than the structured regions (consider the step-edge images of different sizes). Other MEF methods perform equally well except for Mertens09 [2], which is not scale-invariant. Mertens09 employs Laplacian pyramid [24] to avoid unwanted artifacts during fusion. The standard implementation of Laplacian pyramid uses a $5 \times 5$ lowpass filter, which may not eliminate high-frequency information before downsampling (by a factor of two), leading to possible aliasing artifacts across scales. As a result, we may only observe scale-invariance when the image resolutions are related by multipliers of two, which is verified by approxi-
Fig. 6. MEF-Net in comparison with DeepFuse [6] and MEF-Opt [8]. (a) Source sequence “Archway” courtesy of Jianrui Cai. (b) DeepFuse. (c) MEF-Opt. (d) MEF-Net.

Fig. 7. Cross resolution generalization. MEF-Net generalizes well across a wide range of resolutions, which are never seen during training.

TABLE II
AVERAGE MEF-SSIM [9] AND MEF-VIF [42] SCORE OF DIFFERENT MEF METHODS AGAINST MEF-NET ON 90 TEST SEQUENCES COMPUTED AT FULL RESOLUTION. BOTH MEF-SSIM AND MEF-VIF SCORES RANGE FROM 0 TO 1 WITH A HIGHER VALUE INDICATING BETTER PERCEPTUAL QUALITY

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MEF-SSIM</td>
<td>0.923</td>
<td>0.945</td>
<td>0.954</td>
<td>0.958</td>
<td>0.862</td>
<td>0.978</td>
<td>0.964</td>
</tr>
<tr>
<td>MEF-VIF</td>
<td><strong>0.969</strong></td>
<td>0.967</td>
<td>0.956</td>
<td><strong>0.972</strong></td>
<td>0.926</td>
<td>0.952</td>
<td>0.967</td>
</tr>
</tbody>
</table>

3) Computational Complexity and Running Time: We conduct a computational complexity comparison of MEF methods in terms of the number of floating point operations. We assume that the number of input channels is $K$, each of which contains $M$ pixels, and the window size used to compute local statistics is $N^2$. All competing MEF algorithms have a complexity of $O(KMN^2)$, except for MEF-Opt [8] which has a complexity of $O(IMN^2)$, where $I$ is the number of iterations. Ideally, the computation across the channel dimension can be parallelized and the value of $K$ should have little impact on the running time (given sufficient code optimization). Due to the fact that $N^2 \ll M$, the spatial resolution $M$ is the dominant term. MEF-Net enjoys the lowest computational complexity because it restricts most of the computation at a fixed low resolution, while the competing MEF algorithms need to perform all the computation at full resolution.

We compare the running time of MEF-Net with existing MEF methods on input sequences of different spatial resolutions or different numbers of exposures. The testing platform is a computer with an Intel i7-6900K 3.2GHz CPU and an Nvidia Titan X GPU. Mertens09 [2], Li13 [4], SPD-MEF [5], and GGIF [7] utilize the CPU, while DeepFuse [6] and MEF-Opt [8] exploit the GPU. We do not report the running time of DeepFuse on sequences of different numbers of exposures due to its strict input constraint. We reduce the maximum iteration number of MEF-Opt to 100 for the ease of drawing. The results are shown in Fig. 8. On the GPU, MEF-Net takes less than 10 ms to process sequences with resolutions ranging from 512s to 2048s and exposures ranging from three to nine, which is $10 \times$ and $1000 \times$ faster than DeepFuse and SPD-MEF, respectively. More importantly, MEF-Net runs in approximately the same MEF-SSIM scores computed at 512s, 1024s, and 2048s in Fig. 7. By replacing Gaussian filtering with guided filtering, GGIF [7] achieves the desired scale-invariance within the same framework.
approximately constant time in spite of the growing spatial resolution and the number of exposures. On the CPU, MEF-Net is still significantly faster than most MEF methods except for the GPU-mode DeepFuse.

In summary, we have empirically shown that the proposed MEF-Net, characterized by the CAN and the guided filter, trained end-to-end, achieves the three desirable properties—flexibility, speed, and quality—in MEF.

B. Ablation Experiments

We conduct comprehensive ablation experiments to single out the contribution of each individual component in MEF-Net. We first train MEF-Net on low-resolution sequences solely. After training, the guided filter is adopted as a post-processing step to jointly upsample the low-resolution weight maps for final fusion. We then train MEF-Net with the guided filter replaced by the simple bilinear upsample. The MEF-SSIM [9] results are listed in Table III, where we see that integrating upsampling techniques with the preceding CAN for end-to-end training significantly boosts MEF-SSIM. This verifies the power of end-to-end training, where MEF-Net is directly supervised by the high-resolution input sequences. Additional performance gain can be obtained by guided filtering over bilinear upsampling. We also provide a visual demonstration in Fig. 9 and find that guided filtering as post-processing exhibits over-exposure out of the window, while bilinear upsampling trained end-to-end shows black banding artifacts due to the excessively coarse weight maps. Guided filtering trained end-to-end for joint upsampling achieves the best visual quality and is the key component of MEF-Net.

We next evaluate the effect of the input resolutions, depths, and widths of the CAN on the performance of MEF-Net. The depth and width represent the number of convolution layers and the number of feature maps in each intermediate layer, respectively. A shallower CAN implies a smaller receptive field. The results are listed in Table IV, from which we have several interesting observations. First, MEF-SSIM increases with input resolution, depth, and width as expected. Second, by changing the input resolution from 128s to 256s, we observe marginal MEF-SSIM improvement by 0.003. Third, MEF-Net achieves satisfactory performance with a fairly shallow and compact architecture (e.g., with 16 feature maps per layer or a depth of five).

We also assess the role of the regularization parameter $\lambda_a$ and the radius $r$ in the guided filter. $\lambda_a$ controls the smoothness of $A_k$, which is evident in Eq. (6). $r$ also affects its smoothness less directly. A large $\lambda_a$ (or $r$) generates a smooth $W_k$ and may not be good at preserving fine details, leading to a decrease of MEF-SSIM in Table V. A small $\lambda_a$ produces a relatively noisy $W_k$ and may introduce dot artifacts, as shown in Fig. 10. Our default setting achieves the best performance.
TABLE V

<table>
<thead>
<tr>
<th>( \lambda_a )</th>
<th>( 10^{-1} )</th>
<th>( 10^{-2} )</th>
<th>( 10^{-3} )</th>
<th>( 10^{-4} )</th>
<th>( 10^{-5} )</th>
<th>( 10^{-6} )</th>
<th>( 10^{-7} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEF-SSIM</td>
<td>0.961</td>
<td>0.961</td>
<td><strong>0.964</strong></td>
<td>0.962</td>
<td>0.961</td>
<td>0.961</td>
<td>0.961</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( r )</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEF-SSIM</td>
<td><strong>0.964</strong></td>
<td>0.963</td>
<td>0.959</td>
<td>0.956</td>
<td>0.950</td>
</tr>
</tbody>
</table>

Fig. 9. MEF-Net in comparison with its variants. (a) Source sequence “House” courtesy of Tom Mertens. (b) Guided filtering as post-processing. (c) Bilinear upsampling trained end-to-end. (d) MEF-Net (Guided filtering trained end-to-end).

Fig. 10. Emergence of the dot artifacts with a small regularization parameter \( \lambda_a \) in the guided filter. (a) \( \lambda_a = 10^{-8} \). (b) \( \lambda_a = 10^{-4} \) (default).

C. MEF-Net as A Universal MEF Approximator

In this subsection, we exploit the fast speed of MEF-Net and use it as a universal approximator to accelerate existing MEF methods. Specifically, we first apply the target MEF method to our dataset. The generated fused images are considered as the ground truths. We then train MEF-Net on input/output pairs that contain the exposure sequences and the corresponding fused images. The training procedure is the same as Section III-D, except that we optimize a perceptual image quality metric—SSIM [43] in the RGB space. We have also experimented with the mean squared error (MSE) suggested in [14], [19], but obtain inferior approximation accuracy.

Fig. 11 shows the visual results of MEF-Net in approximating SPD-MEF [5] and GGIF [7] on the source sequence “Stone house”. Although the two MEF methods produce different overall appearances, MEF-Net is able to closely match them. On the 90 test sequences, the approximation accuracy in terms of SSIM for SPD-MEF and GGIF is 0.961 and 0.976, respectively, demonstrating the promise of MEF-Net as a universal MEF approximator. On sequences of resolution 1024\( \times \)s, we speed up SPD-MEF and GGIF more than 1000 and 100 times, respectively.

V. Conclusion and Discussion

We have introduced MEF-Net, a fast MEF method based on deep guided learning. The core idea of MEF-Net is to predict the low-resolution weight maps using a CAN and jointly upsample them with a guided filter for final weighted fusion. The high speed of MEF-Net is achieved by restricting the main computation at a fixed low resolution and parallelizing the computation across exposures. The visual improvement of the fused images is achieved by end-to-end training with MEF-SSIM measured at full resolution. In addition, we demonstrate the promise of MEF-Net as a universal MEF approximator to accelerate existing and future MEF methods.

The current MEF-Net works with static scenes only. How to extend it to account for dynamic scenes is an interesting and challenging problem yet to be explored. The major impediment here is the lack of perceptual image quality metrics for dynamic scenes or ground truths for supervision. Kalantari and Ramamoorthi [16] put substantial effort in capturing static and dynamic exposure brackets of the same scene and treated the static sequences as a form of ground truths. Cai et al. [20] made use of 13 existing MEF and HDR deghosting methods to generate a set of candidates and manually picked the best ones as the ground truths. Both processes are expensive and time-consuming, which limit the number of collected sequences and hinder the generalizability of the learned convolutional.
Developing flexible and fast MEF methods that are able to handle dynamic scenes is desirable and worth future investigations.

ACKNOWLEDGEMENTS

The authors would like to thank Jianrui Cai, Yuki Endo, K Ram Prabhakar, Dr. Nima Kalantari and Dr. Mark Fairchild for providing their multi-exposure sequences, and Hui Li for fruitful discussions. We thank the NVIDIA Corporation for donating a GPU for this research.

REFERENCES


