Vision, Modeling, and Visualization (2019), pp. 1–3
H.-J. Schulz, M. Teschner, and M. Wimmer (Eds.)

Reconfigurable Snapshot HDR Imaging Using Coded Masks and Inception Network Supplementary

M. Alghamdi, Q. Fu, A. Thabit & W. Heidrich

King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Saudi Arabia

1. Calibration Network

Our full calibration network is depicted in Figure 1. Given $N = 15$ randomly captured raw CFA coded exposure images, our network learns an estimate $\hat{\Phi}$ of the original mask Φ. Directly estimating $\hat{\Phi}$ is a challenging task, to help our network converge, we regularize the process by making the network also learn to demosaick the green channel of all N images. We jointly minimize the ℓ_2-loss of both $\hat{\Phi}$ and $\hat{G}$ defined in Eq.(1). We add a regularizing parameter μ to control the effect of our green channel demosaicking.

$$\begin{aligned}\ell\left(\Phi,\hat{\Phi},G,\hat{G}\right) = \frac{1}{WH}\sum_{j=1}^{H}\sum_{i=1}^{W}\left|\Phi(i,j)-\hat{\Phi}(i,j)\right|^2 \\ +\mu\left|G(i,j)-\hat{G}(i,j)\right|^2.\end{aligned} \tag{1}$$

During training phase we experiments with N= 5, 10 and 15, table 1 show the training loss as defined in eq. 1. Clearly larger N leads to lower training error, however it will increase the number of network parameters and memory usage.

Table 1: Calibration training loss for different N. Larger N value have a better training loss

N	5	10	15
Training loss	0.050	0.006	0.002

2. Calibration Results

Our calibration network first learns the mask from a series of randomly selected LDR images. For a total of 260 images in the testing set we randomly select 15 HDR images, re-size them to 768×1024, and perform 90th percentile normalization as in eq. 2. We select a camera model from the 67 camera models that we trained for (for more information refer to section 3.4 of the main paper). We simulate the corresponding 15 coded LDR images given a ground truth mask, which has a continuous uniform random distribution in the range $[0,1]$. The exposure time is fixed for most of the images as a baseline, where we set the exposure time depends on the chosen camera , and the normalized HDR scenes such that most of the images are properly exposed (the number of saturated and under-exposed pixels are minimized). While 3 or 4 of which are randomly chosen to have 1 or 2 stops more exposure. Such an operation facilitates the regions with low values in the mask. We also take into account the Bayer color filters in the simulated LDR images. We input the 15 simulated coded LDR images to the calibration network, and the network outputs the learned mask, as well as the demosaicked green channel of the LDR image. In Figure 2 we show an example of our calibration results. We repeat the previous test for 260 times the measurements mean absolute error $(MAE = 0.037)$, mean squared error $(MSE = 0.002)$, and the standard deviation $(STD = 0.03)$.

In real experiments, our calibration network works well in common illuminations, e.g. sunlight, indoor fluorescent light, tungsten light etc. The camera is set without any gain to avoid noise amplification. In addition, there should be no large overexposed areas in the calibration images.

3. Dataset Preprocessing

We divide the HDR Core into 910 training and 260 testing images. It is common practice in deep learning to normalize input data within a constant range. In our case, we use the 90th percentile normalization defined in (2), where $l_{i,c}$ is the i-th pixel in color channel c, ρ is the target scalar for the 90th percentile of each images, and p_{90} is the 90th percentile of in the original image.

$$H\left(l_{i,c}\right) = l_{i,c} \times \frac{\rho}{p_{90}}. \tag{2}$$

Our next step is to simulate the mask encoding. First, we modulate light arriving at the sensor by multiplying the radiance values of the input augmented HDR image by a randomly generated transmittance mask. Then we use a noise model proposed by Konnik and Welsh [KW14] to simulate various realistic noises, coded and filtered by the Bayer filters to obtain the raw image. For the MIT Places dataset and the HD YouTube frames we follow exactly the same preprocessing.

We prepare the training data offline. We use the full resolution images from the MIT Places dataset to avoid ringing artifacts resulting from downsampling. We encode each image with a random mask, apply the noise simulation and then crop 16 random patches of size $128 \times 128 \times 3$. Each image from the MIT Places dataset is

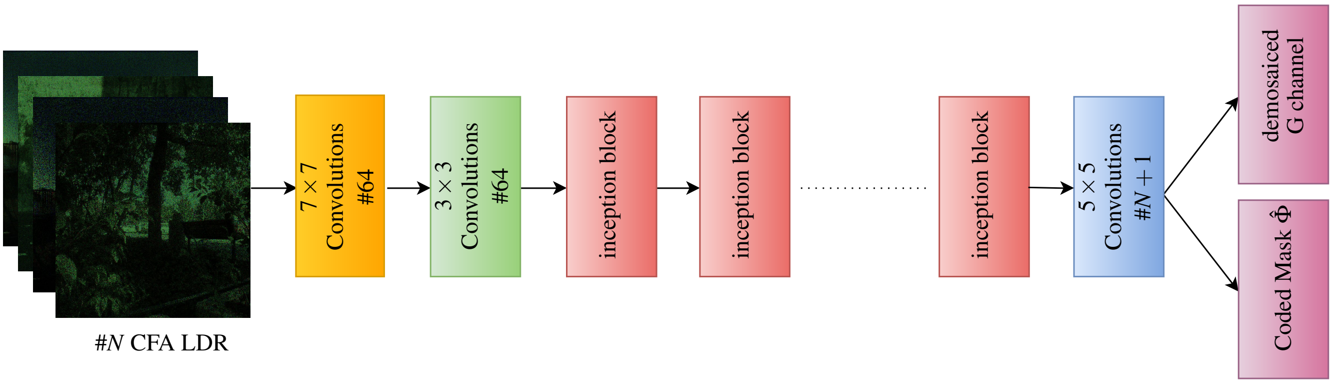

Figure 1: Calibration inception network. Given $N = 15$ raw CFA coded exposure images, our network learns an estimate $\hat{\Phi}$ of the original mask Φ. Our network consists of 2 convolutional layers followed by 8 inception blocks (illustrated in main paper) and followed by a convolutional layer. The numbers of filters are indicated in each convolution layer in our implementation.

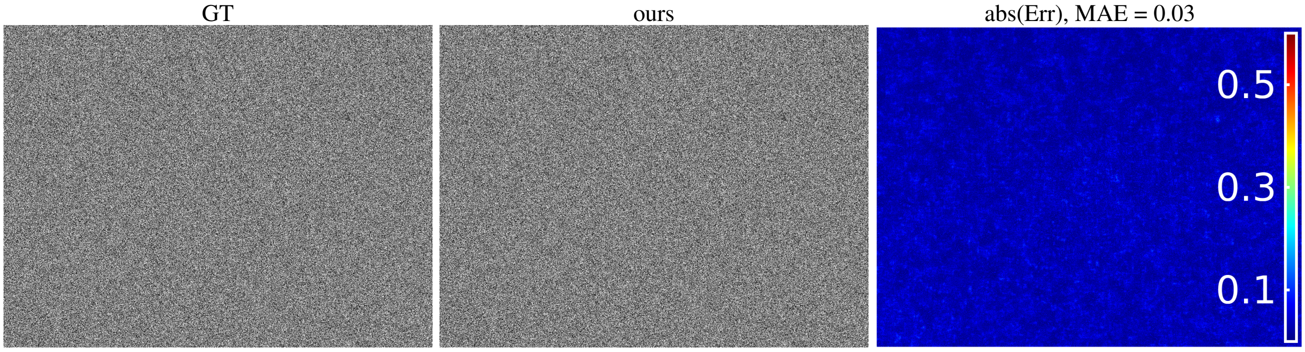

Figure 2: An example of simulated calibration results by our trained network. Left: ground truth mask, Middle: network estimated mask. right: absolute error.

augmented with one random Gaussian mask and one random uniform mask, resulting in $40M$ crops for both types of masks. For the YouTube HD frames we follow the same processing, however, each image is augmented with two random unifrom mask and two LF Gaussian mask, resulting in $1.2M$ crops.

For the HDR core training set we use the 910 samples to generate $120K$ coded LDR images (16 random patches of size 128×128 are cropped randomly in each coded image). For each image we randomly select a camera model, set the exposure, and then randomly select a random mask that corresponds to a certain distance between the binary mask and the sensor, which controls statistics of the mask pattern.

3.1. Hyperparameter Search

We fix the parameters of all the inception blocks in our implementation, such that the input and output of each inception block has 64 channels.

We use the Xavier uniform initializer for all weights [GB10]. For the loss function parameter λ we examine the following values, λ = [0.1, 0.3, 0.5, 0.7, 0.9].

For the L_{VGG} loss, we experiment using the features from VGG_{19} [SZ14]:

- Last convolutional layer before the first max-pooling layer ‘conv1-2’.
- Last convolutional layer of VGG_{19} before the last max-pooling layer ‘conv5-4’.
- All the convolutional layer proceeding max-pooling layers (five in total).

For mask calibration we use a total number of $N = 15$ randomly captured raw coded LDR images as inputs to the network. For learning we use the ADAM optimizer [KB14]. We set a learning rate for all networks to 10^{-5}. We use the Keras API on top of TensorFlow [C*15]. We train and test all the networks using two GeForce GTX GPU. After the training is finished, our HDR reconstruction network runs on both real and synthetic data.

3.2. Pre-Training and Fine-Tuning

We train our network in three stages. First on the MIT Places dataset we use the ℓ_2 norm. We train this network for around 3M gradient updates. Then we use the learned wights to train on the

YouTube HD (16-bit) dataset with the following loss function:

$$L_{HDR}(x,\hat{x}) = \lambda L_1(x,\hat{x}) + (1-\lambda)L_{VGG}(x,\hat{x}). \qquad (3)$$

For this stage we train for different λ values and different VGG_{19} features as mentioned above for 2M steps. After that we compute the HDR-VDP Q_{score} for all models on the YouTube HD data set. The highest HDR-VDP Qscore is obtained with $\lambda = 0.3$, and by using all the convolutional layer proceeding max-pooling layers (five in total) of VGG_{19}. The best model is fine-tuned on HDR-Core dataset for another 2.5M steps. For training we used patch size of 16 on the Places dataset. For HDR-Core dataset and YouTube HD data set we set the patch size to 4.

References

[C*15] CHOLLET F., ET AL.: Keras: Deep learning library for theano and tensorflow., 2015. URL: https://keras.io. 2

[GB10] GLOROT X., BENGIO Y.: Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), pp. 249–256. 2

[KB14] KINGMA D. P., BA J.: ADAM: A method for stochastic optimization. *arXiv preprint* (2014). arXiv:1412.6980. 2

[KW14] KONNIK M., WELSH J.: High-level numerical simulations of noise in CCD and CMOS photosensors: review and tutorial. *arXiv preprint* (2014). arXiv:1412.4031. 1

[SZ14] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *arXiv preprint* (2014). arXiv:1409.1556. 2