Sensor-realistic Synthetic Data Engine for Multi-frame High Dynamic Range Photography

Jinhan Hu*[1], Gyeongmin Choe[2], Zeeshan Nadir[2], Osama Nabil*[3], Seok-Jun Lee[2], Hamid Sheikh[2], Youngjun Yoo[2], and Michael Polley[2]

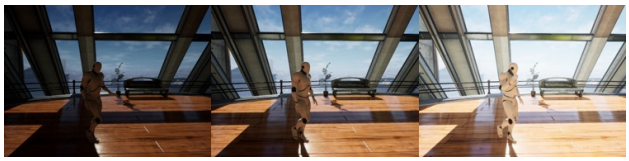
[1]Arizona State University, [2]Samsung Research America, [3]UCLA

Abstract

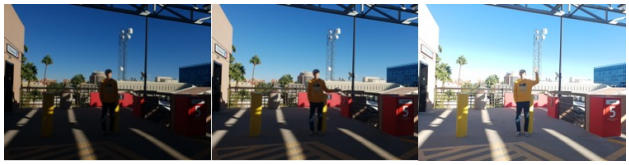
Deep learning-based mobile imaging applications are often limited by the lack of training data. To this end, researchers have resorted to using synthetic training data. However, pure synthetic data does not accurately mimic the distribution of the real data. To improve the utility of synthetic data, we present a systematic pipeline that takes synthetic data coming purely from a game engine and then produces synthetic data with real sensor characteristics such as noise and color gamut. We validate the utility of our sensor-realistic synthetic data for multi-frame high dynamic range (HDR) photography using a Samsung Galaxy S10 Plus smartphone. The result of training two baseline neural networks using our sensor realistic synthetic data modeled for the S10 Plus show that our sensor realistic synthetic data improves the quality of HDR photography on the modeled device. The synthetic dataset is publicly available at https://github.com/nadir-zeeshan/sensor-realistic-synthetic-data.

1. Introduction

With the recent breakthrough in deep learning and the growth in computational capability of mobile phone platforms, deep learning-based imaging applications have been extensively deployed on mobile devices [54, 9]. Such applications include synthetic aperture (bokeh) [48], night shot [5], burst shot [1], etc. However, one generic trained network model does not fit for all mobile devices, as different mobile devices have different cameras (image sensors) with unique characteristics. Thus, the network model needs to be tuned for an individual device in order to maximize its capability [9]. An important requirement to train deep learning frameworks for developing an accurate model is the availability of abundant training data [18, 7]. Capturing a large training dataset for training deep learning models for specific sensors is nearly impossible in the real-world because real data collection is limited by several constraints, including cost and privacy [24]. While there has been progress made on the computational fronts, the deep learning community still has a "data problem" at large.



(a) One example of dynamic sets in our synthetic training data at three exposure levels (i.e., low, medium, and high from left to right).



(b) One example of real images captured by a smartphone at three exposure levels (i.e., low, medium, and high from left to right).

Figure 1. Our synthetic dataset for multi-frame HDR contains diverse 3D background models and various subject motions to simulate dynamic real-world HDR scenarios.

The need for training data has led to an interest in synthetic data. Synthetic data offers many obvious advantages over real data: *(i)* There is no privacy concern in synthetic data, as people do not have to worry about their personal information being exposed. *(ii)* Synthetic data is cheap at large-scale; ideally, researchers can generate an infinite amount of data. *(iii)* In many cases, it is easy to generate ground-truth for synthetic data. *(iv)* Synthetic data is more flexible to be customized, e.g., motions can be customized for the deghosting problem in HDR [46, 21]. One approach to generate synthetic data is using the game devel-

*Works done while interns at Samsung Research America MPI lab. Contact the primary author Jinhan Hu at jinhanhu@asu.edu.

opment platforms, such as the Unreal Engine [15]. However, a key shortcoming of such synthetic data is that it does not model actual camera characteristics, such as color space and noise. The gap between synthetic and real data may not pose a huge difficulty in some applications, such as optical flow [3, 11]. However, for other applications, the gap between synthetic and real data presents issues; for an HDR imaging task, learning the color, tone and noise characteristics of the training data is important for accurate results.

HDR is one of the most essential imaging applications for smartphone cameras to capture high quality and well-exposed pictures. Multi-frame HDR increases the dynamic range of captured images by merging several low dynamic range (LDR) frames of different exposure levels. Merging suffers from corner cases and artifacts from dynamic scenes (objects moving in scenes) [51]. Researchers have introduced several deep learning-based HDR imaging approaches to overcome the limitations. Kalantari *et al*. [26] proposed to learn the merging process. Wu *et al*. [52] further proposed an end-to-end deep learning-based HDR solution to handle large foreground motions in consecutive captures without using optical flow for image registration. However, because of the limitations of capturing real data, both prior works only compiled and used a small training dataset (around 75 sets). Moreover, pre-trained models from these works yield poor quantitative and qualitative HDR results, as we tested on smartphone images captured by a Samsung Galaxy S10 Plus smartphone.

As far as we are aware of, no prior work has used synthetic data in training deep learning models for HDR imaging because using synthetic data without modeling sensor characteristics in HDR applications sometimes lead to bad results. Figure 2 shows that fine-tuning a HDR network (pre-trained on real data) with pure synthetic data (without modeling the sensor color space and noise characteristics) makes the HDR network more susceptible to noise and produces worse HDR outputs. Therefore, the gap between real and synthetic data such as noise and color space needs to be filled before the synthetic data can be used reliably for deep-learning based HDR imaging.

To improve the utility of synthetic data, we explore a pipeline to apply sensor realism to pure synthetic data by carefully modeling the sensor's color space and noise characteristics. We first generate pure synthetic data from the Unreal Engine [15]. Then, we use the targeted image sensor to capture a few real images containing a color checker chart [53]. After the color and noise characteristics are extracted from those images, we apply those characteristics to the engine-generated pure synthetic data, such that sensor-realistic data tailored to the targeted image sensor is obtained. Currently, our HDR dataset contains 150 sets of pure synthetic data and 150 sets of sensor-realistic synthetic data (modeled for the S10 Plus). Notice that each sensor has its unique realism. The color calibration and noise modeling pipeline needs to be performed for each different sensor separately.

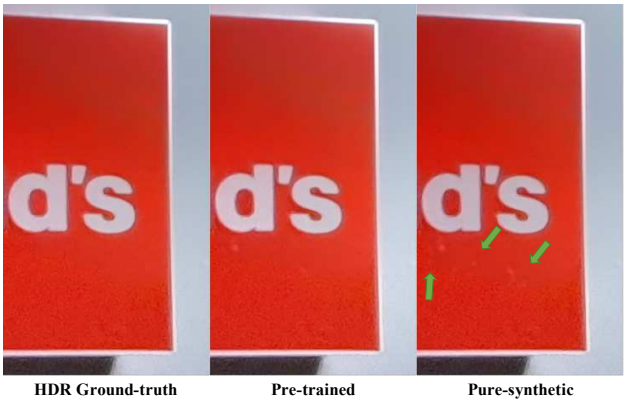


Figure 2. Fine-tuning a HDR network (pre-trained on real data) with pure synthetic data (without sensor modeling) makes the network more susceptible to noise. Excessive unnatural white dots can be identified in the output generated from the network fine-tuned with pure synthetic data (pure-synthetic) than the network pre-trained on real data (pre-trained).

We validate our sensor-realistic synthetic dataset through training on two baseline deep HDR networks. Both networks follow an encoder-decoder based architecture [36]. Network-1 is pre-trained with real images captured by our targeted S10 Plus. Network-2 is presented by Wu *et al*. in [52], which is pre-trained using the Kalantari dataset [26] captured by a Canon EOS-5D Mark III camera. We test both networks on static scenes captured by our targeted S10 Plus, even though network-2 is pre-trained on dynamic scenes captured by a different device. In this work, we focus on analyzing the utility of sensor-realistic synthetic data for improving learning-based HDR imaging on the modeled device. However, we will explore how to utilize synthetic data (e.g., engine-generated accurate motion masks) to handle dynamic scenes in future works. Results show that both networks yield improved HDR results generated from real images captured by the S10 Plus after being fine-tuned with our sensor-realistic synthetic dataset modeled for that S10 Plus. In addition, we apply different sensor models to the synthetic data (two color spaces and three noise levels) to analyze their influence on real HDR output. The results confirm that applying the correct sensor model is necessary to yield the best HDR output.

Our main contributions are summarized as follows:

- We solve the problem of insufficient training data in deep learning-based HDR imaging by introducing a synthetic data generation pipeline which produces sensor-realistic synthetic data that has the color and noise characteristics of the modeled image sensors.
- We contribute a multi-exposure sensor-realistic syn-

thetic dataset modeled for a Samsung Galaxy S10 Plus (examples shown in Figure 1). The generated sensor-realistic synthetic data is a practical solution to improve the performance of deep learning-based HDR imaging for specific mobile devices.

- We present a detailed analysis on applying different sensor realism models to synthetic data for affecting the performance of the real HDR output.

2. Related Work

AI HDR imaging Kalantari *et al*. [26] introduced learning-based multi-frame HDR merging. Wu *et al*. [52] further proposed an end-to-end deep learning-based HDR framework without using optical flow for image registration. Eilertsen *et al*. [12] proposed a deep learning algorithm to reconstruct an HDR image from a single exposure. Zhang *et al*. [58] proposed to learn to reconstruct HDR from LDR panoramas. Yang *et al*. [56] designed a novel framework in which they utilized one CNN for reconstructing missing details in the HDR domain in LDR images and another CNN for tonemapping on the predicted HDR details. Yan *et al*. [55] introduced an attention-guided network to improve HDR merging by reducing errors caused by misalignment and saturation. Mostafavi *et al*. [23] combined GAN and event cameras to generate HDR images even in extreme illumination conditions. Though many previous works tried to improve HDR imaging by leveraging the power of neural networks, none of them attempted to integrate synthetic data into the training process.

Synthetic data for AI tasks Synthetic data is already broadly deployed for training neural networks in a variety of tasks, including pose estimation [47, 34], object detection and tracking [14, 20], text localization [16], semantic segmentation [17, 22, 45, 29], generating X-Ray images [41], etc [49]. Other than public synthetic datasets such as Sintel [4], SYNTHIA [37], Flying Chairs [11], and Virtual Kitti [14], some researchers have tried to design frameworks and tools to help others generate good synthetic data from multiple sources [31, 44, 33, 35, 59]. However, the synthetic data generated from these sources does not appear realistic. To address this, some researchers have tried to improve the realism of synthetic data from multiple aspects to better serve neural networks [38, 40, 28, 39, 2, 43, 17, 45]. Shrivastava *et al*. [40] introduced simulated and unsupervised learning to improve the realism of synthetic eyes for better performance on gaze estimation and hand pose estimation. Tremblay *et al*. [43] improved the realism of synthetic data by randomizing its textures, pose, lighting, etc. Ledig *et al*. [28] added a perceptual loss function into the training process to recover more realism in the generated super-resolution images. However, none of these works modeled the realism of different sensors, such as the sensor specific noise characteristics and color gamut. Thus, these works cannot produce synthetic data that is realistic enough to train the network to yield the best result on targeted devices, especially for data-driven computational photography tasks such as deep HDR imaging.

3. Sensor-realistic Synthetic Data

Figure 3 shows our proposed sensor-realistic synthetic data generation and processing pipeline together with one example of the synthetic data before and after sensor realism is applied. Unreal Engine is used for rendering various synthetic models and motions and generating synthetic multi-exposure images. Then, to bridge the gap between synthetic and real data, we model the sensor realism (color calibration and noise modeling) of the targeted sensor in accordance with real capture parameters, such as exposure time and ISO. Sensor-realistic synthetic data helps neural networks to better deal with color and noise which are often very important factors in real images. Please refer to the supplementary material for steps about how to implement this pipeline to get your sensor-realistic synthetic data tailored for your targeted mobile device.

3.1. Synthetic Data Generation

Unreal Engine is a suite of tools for game developers to design and build games [15]. This engine has been proven effective in rendering a variety of synthetic scenes. Our work utilizes various Unreal Engine human models including a variety of indoor and outdoor 3D environments, as well as various types of human models with diverse motions to simulate realistic data examples.

We implement a synthetic data capture loop using the Blueprints visual scripting system in Unreal Engine. In each synthetic data capture loop, we move the virtual camera to a desired location and then press the capture button (e.g., a keyboard input), just like capturing images in real-world. In particular, we are able to control the following factors in capturing synthetic data. First, we can control the camera view point to precisely control the under- and over-exposed regions inside each image. By doing this, we can simulate diverse HDR scenarios which are hard to acquire when capturing real images. Second, we can control the subject motion, e.g., the movement of a human models. In particular, we stop the subject motion to generate static sets for the purpose of acquiring ground-truths. Then, we resume the subject motion to generate dynamic sets which can simulate challenging scenarios containing possible motions in real-world HDR captures. Third, we can control the camera parameters to generate images at different exposure levels (e.g., {EV-2, EV0, and EV2}). This capture loop can be repeated until enough training data is acquired.

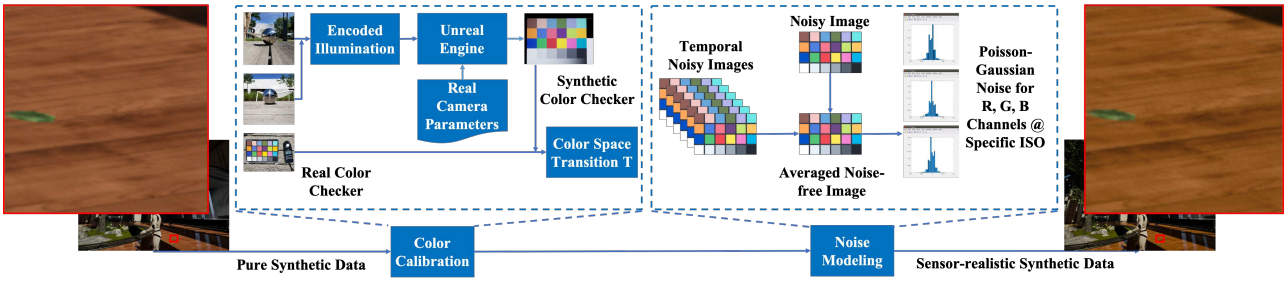


Figure 3. Our sensor-realistic synthetic data generation and processing pipeline generates synthetic multi-exposure images, models the sensor realism (color calibration and noise modeling), and applies the sensor realism to pure synthetic data to turn it into sensor-realistic synthetic data. This sensor-realistic synthetic data example shows warmer color tones similar to the modeled device.

3.2. Color Calibration

The color gamut [50] of real cameras differ from manufacturer to manufacturer and model to model. Therefore, the captured images yield different color and tone characteristics. However, despite Unreal Engine producing colorful images, it always generates data in the same color space that does not necessarily resemble with the sensor of interest. The color calibration component in Figure 3 shows our approach to model the color space difference between Unreal Engine and the real sensor.

To begin with, we put a light probe inside a targeted scene [8, 32]. The light probe we used is a stainless steel gazing ball. It helps to provide the lighting information for a static scene. Then, we use the modeled image sensor (a S10 Plus in this work) to capture two images of that light probe from two different angles. The two captured images contain all illuminations in 360° of that targeted scene. After that, we use the HDRShop [27] tool to automatically encode the illumination information extracted from the two captured images. Then, the illumination can be imported into Unreal Engine as a cubemap to light up a synthetic color checker. We adjust the exposure, exposure compensation, ISO, aperture, and light intensity in the *PostProcessingVolume* [15] for the virtual camera inside Unreal Engine according to the camera parameters in real captures. Finally, we proceed to capture the synthetic color checker ($Color_{Synthetic}$) with the virtual camera. To acquire the real color checker image, we simply use the modeled image sensor again to capture a real color checker ($Color_{Real}$) in the targeted scene under the same lighting condition. By sampling the color difference between the real color checker $Color_{Real}$ and the synthetic checker $Color_{Synthetic}$, we estimate the color space transition matrix T using least squares for the modeled device, as shown in Equation 1.

$$Color_{Real} = T \cdot Color_{Synthetic} \tag{1}$$

3.3. Noise Modeling

To mimic real smartphone images, we model the smartphone camera sensor noise characteristics using normal approximation to Poisson-Gaussian noise [57, 25, 13]. To validate this approximation, we conduct experimental analysis of the sensor noise in raw images. A total of 600 images were captured under twelve imaging conditions. Each imaging condition has a different combination of illumination temperature, ISO level and illumination brightness. For each imaging condition, we captured 50 frames to obtain a noise-free image by time-averaging after excluding defective and saturated pixels. From this dataset, we estimate the parameters (a, b) of a signal-dependent additive Gaussian noise with the distribution given as:

$$\mathcal{N}(z|0, \sigma(i,j)^2) = \frac{1}{\sigma(i,j)\sqrt{2\pi}} e^{\frac{-z^2}{2\sigma(i,j)^2}} , \tag{2}$$

where $\sigma(i,j)^2$ is the pixel dependent variance given as

$$\sigma(i,j)^2 = a * I(i,j) + b , \tag{3}$$

and $I(i,j)$ is the signal value at pixel (i,j). The parameters a and b control the amount of signal dependent and signal independent components of noise estimated for different ISO levels, respectively.

4. Network Models

To validate our sensor-realistic synthetic dataset, we train two different AI networks for multi-frame HDR imaging. First, we use the basic encoder-decoder based architecture, also known as U-Net. We use this architecture to generate the blending weights (Blendmap) for the input images [26] and then use the exposure fusion algorithm to blend the input frames using the blending weights obtained through the network [30]. Second, we use the network presented in [52], which is a derivative of an encoder-decoder architecture. This network takes several LDR images at different exposure levels and directly outputs the HDR image.

Besides the architectural difference, Network-1 is pretrained on a real dataset collected by us using a Samsung Galaxy S10 Plus smartphone. Whereas, Network-2 is pretrained on the dataset provided by [26] captured by a Canon EOS-5D Mark III camera. We are not comparing the HDR

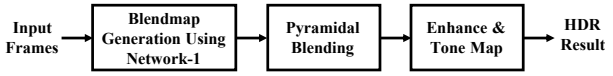


Figure 4. HDR image processing pipeline in Network-1 replaces the Blendmap Generation step with the deep learning network to produce the blending weight maps.

quality between these two networks. However, we focus on demonstrating the effectiveness of our sensor-realistic synthetic data by observing improvements in HDR results for both networks.

4.1. Network-1

The first network we use is U-Net, which is famous in image segmentation applications [10, 36, 6]. We pose the HDR imaging as a segmentation problem such that our network produces a discrete blending weight for every pixel. We discretize the blending weights into 256 levels such that for a pixel i, the blending weight $w_i \in \{\frac{0}{256}, \frac{1}{256}, \ldots, \frac{256}{256}\}$. To generate the final HDR output from the blending weights, we use Laplacian pyramids of the input images and blend them similar to the exposure fusion algorithm [30]. To tone map the output image, we apply an exponential tone mapping curve, similar to gamma correction. Figure 4 shows our HDR image processing pipeline for this network.

The network architecture shown in Figure 5 contains an encoder branch, as well as a decoder branch. The encoder branch of the network helps gain contextual knowledge of the input images to understand the general image structure, whereas the decoder branch reconstructs the final blending weight maps. Between the encoder and the decoder branches, there are skip connections that help in increasing the resolution of the blending weight maps.

We use a weighted L1 loss function to train the model considering that the training data is unbalanced, i.e., it contains unequal amounts of different tonal values. When computing the weighted L1 loss function, we penalize the loss function for each of the 256 output channels differently. The weighting used for each of the 256 channels is inversely proportional to their frequency of occurrence in the training data. Mathematically, the loss function is given as follows:

$$L(x, y) = \sum_{w \in W} \sum_{i=0}^{255} \left| f_w\left(x, \theta\right) - y_w \right| \alpha_i \, I\left(y_w == i\right), \quad (4)$$

where θ are network parameters, x are input image frames, $f_w(x, \theta)$ is network output on pixel w, W is the patch, α_i is weighting for output channel i and $I(.)$ is indicator function. Figure 6 shows the weighting coefficients computed for both real and synthetic training data. We use the respective weighting coefficients when training the network.

To pre-train this network, we use 308 static scenes acquired from a Samsung Galaxy S10 Plus smartphone. For

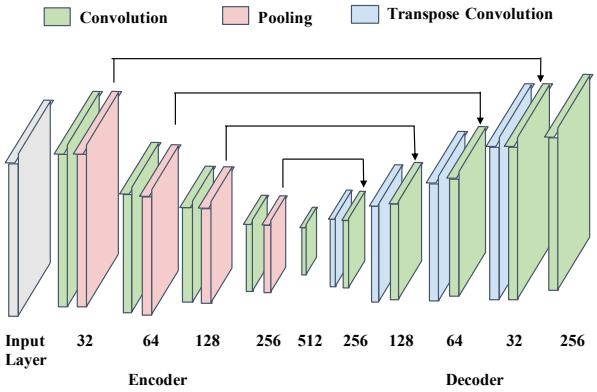
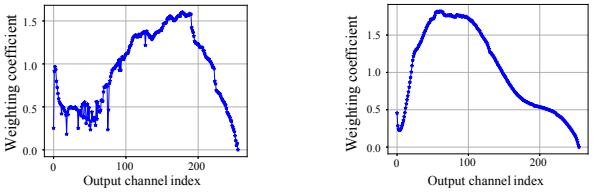


Figure 5. The architecture used to produce blending weight maps in Network-1. Each block has their number of channels written underneath them. All encoder blocks consist of two convolutional layers. The decoder blocks consist of a transposed convolutional layer followed by two convolutional layers. We use 3×3 filters for all the layers except for the last layer where we use 5×5 filters.



(a) Weighting coefficient for real training data.

(b) Weighting coefficient for synthetic training data.

Figure 6. Weighting coefficients computed over the two different training datasets. Weighting coefficients are inversely proportional to the frequency of occurrence of different classes in the training data ground truth blending maps. Class 0 represents fully saturated region, whereas class 255 represents fully underexposed region.

each scene, we use a pair of low exposure and high exposure frames in the 8-bit RGB format. These frames are first fed to the exposure fusion algorithm [30] to generate the blending weights, which is used as the ground-truth to pre-train the network. The pre-trained network produces the baseline results for comparisons. Finally, we fine-tune this network using 150 static samples of sensor-realistic training data.

4.2. Network-2

Second, we use the network architecture proposed by Wu *et al.* in [52]. The network architecture is a derivative of U-Net. There are three different encoder branches, one for each of the three input exposure levels, a merger branch, and a decoder branch. The loss function is computed using the L2 norm, however, the network output and the ground-truth images are tone-mapped first using μ-law with $\mu = 5000$ [52].

In our experiments, we use the pre-trained network pro-

vided by the authors, which is trained on Canon EOS-5D Mark III camera for producing the baseline results. To train the network with our sensor-realistic synthetic data, we use the same loss function and training scheme as [52]. More specifically, for each sample, we first generate a set of static frames at three different exposure levels as well as a set of dynamic frames at the same three exposure biases. Next, we apply the color space and noise model on these frames to model the sensor characteristics. The middle exposure level is treated as the reference frame. The network learns to handle the motion by manipulating the training data. During the training, the two non-reference frames in the static set are replaced by the corresponding dynamic frames and this new set of three frames makes a single training sample [26, 52].

5. Dataset

Except for the real datasets used to pre-train the networks, we use both synthetic and real datasets in the experiments performed in §6. The real dataset is used to test the final HDR results, whereas the synthetic dataset is used to fine-tune the network models pre-trained on real data. Notice that because of the many differences between Network-1 and Network-2, such as the exposure level of input images and the combinations of real data used in pre-training, we used 1) different versions of synthetic dataset in fine-tuning each network (static data for Network-1 and dynamic data for Network-2); 2) different methods to generate ground-truth (a proprietary algorithm for Network-1 and the linear triangular method for Network-2). However, both networks are tested on static real datasets because we are interested in exploring the utility of synthetic data to increase the dynamic range of the photograph without having a confounding effect coming from ghost artifacts. We intend to explore deghosting with the help of synthetic data in a future paper.

Datasets used in Network-1 A pure synthetic dataset contains 150 sets of static synthetic images generated by following the procedure in §3.1. A sensor-realistic synthetic dataset contains 150 sets of static synthetic images with the sensor realism modeled for a Samsung Galaxy S10 Plus smartphone applied by following the procedure described in §3.2 and §3.3. A real testset (testset-1) contains 24 sets of static images captured by the modeled device at a resolution of 4032x3024. All three datasets are captured at two exposure levels (i.e., {EV-3 and EV0}). We use a proprietary algorithm to compute the Blendmap first and then use the exposure fusion algorithm [30] to generate the HDR ground-truth for datasets used in this network.

Datasets used in Network-2 We used four versions of sensor-realistic synthetic datasets each containing 150 sets of synthetic images with different color calibration models and noise models applied to fine-tune Network-2. Each set in each synthetic dataset contains a static subset with motion stopped and a dynamic subset with motion resumed. A real testset (testset-2) contains 15 sets of static images captured by our modeled device at a resolution of 4032x3024. All datasets are captured at three exposure levels (i.e., {EV-2, EV0, and EV2}). We follow the same linear triangular method described in [52] to generate the HDR ground-truth from the static sets, replace the reference frame in the dynamic sets by the reference frame in static sets, and then feed the modified dynamic sets together with the ground-truth to the network for fine-tuning.

Please refer to the supplementary material for some examples in each synthetic dataset.

6. Evaluation

6.1. Experiments

We evaluate and verify the effectiveness of our sensor-realistic synthetic data in three separate experiments.

1. We fine-tune Network-1 with the pure synthetic dataset and the sensor-realistic synthetic dataset, separately. Then, we compare the HDR results quantitatively among the original pre-trained network (Net_O), the network fine-tuned with pure synthetic data (Net_P), and the network fine-tuned with sensor-realistic synthetic data (Net_S), tested on the real testing dataset testset-1.

2. We fine-tune Network-2 only with the sensor-realistic synthetic dataset. We compare the HDR results quantitatively among Net_O and Net_S, tested on the real testing dataset testset-2.

3. We fine-tune Network-2 with different versions of the sensor-realistic synthetic dataset with different sensor models (two color models and three noise levels) applied. We analyze in detail how different sensor models will affect the quality of the real HDR output, also tested on testset-2.

To train the network, we use the TensorFlow library [42] on the Linux platform in all three experiments. In Experiment 1, we train Network-1 for 300 epochs on two Nvidia Titan Xp GP102 GPUs. To augment the data, we select a patch randomly in the input samples, and perform random horizontal and vertical flips as well as rotations by 90 degrees. In Experiment 2 and 3, we train Network-2 presented in [52] for 50 epochs on an Nvidia GeForce RTX 2080Ti GPU. Data augmentation (random flipping and rotating) is performed while generating TFRecords [42] before training, as introduced in [52].

6.2. Quantitative Results

To evaluate and compare the results, we quantify the HDR output using metrics mentioned in [26]. We com-

Network-1	Net_O	Net_P	Net_S
PSNR-T	40.40	39.48	41.56
PSNR-L	37.70	37.06	39.54
SSIM-T	0.9927	0.9893	0.9930
SSIM-L	0.9934	0.9880	0.9937

Table 1. Network-1 fine-tuned with our sensor-realistic synthetic dataset (Net_S) outperforms the network pre-trained with real data only (Net_O), and the network fine-tuned with pure synthetic data (Net_P). The results are averaged across 24 samples in testset-1.

pute PSNR-T for the peak signal-to-noise ratio between two tone-mapped images and PSNR-L for the peak signal-to-noise ratio between two images in the linear domain. Similarly, we compute the perceptual difference SSIM-T after tone mapping and SSIM-L in the linear domain. All quantitative results are averaged across the number of test images.

Table 1 shows the HDR results from experiments performed on Network-1 (Experiment 1). The network fine-tuned with our sensor-realistic data yields the best HDR results in all metrics, whereas the network fine-tuned with pure synthetic data actually performs worse than the network pre-trained with real data only. Figure 7 shows an example of the HDR results before and after the network is fine-tuned with our sensor-realistic synthetic dataset. In detail, the network fine-tuned with the sensor-realistic synthetic dataset outputs more details on the background through the window. Note that the improvement coming from the sensor-realistic synthetic data is not very drastic in Network-1 because this network is already pre-trained with real data captured by the modeled device (the same sensor realism model applied to synthetic data).

Table 2 shows the HDR results from experiments performed on Network-2 (Experiment 2). This network was pre-trained on Kalantari dataset coming from Canon EOS-5D Mark III camera [26]. The pre-trained network has very low PSNR-T and SSIM-T values when tested on Samsung Galaxy S10 plus images from testset-2. However, after fine-tuning with our sensor-realistic synthetic dataset, PSNR-T and SSIM-T are improved substantially by 26.82% and 34.29% respectively, tested again on testset-2 captures coming from the modeled device. Figure 8 shows an example of the HDR output before and after the network is fine-tuned with our sensor-realistic synthetic dataset. Note that the images are equally enhanced using the Photomatix tool [19] for visualization. In Figure 8, the network outputs better detail in the window and tree. In Experiment 2, we did not fine-tune Network-2 with pure synthetic data because this network is pre-trained with a completely different dataset captured by a Canon camera. Fine-tuning the network with sensor-realistic data after pre-training the network with data coming from a different sensor can have confounding effects because of the non-convexity of the optimization problem and the performance can be unpredictable. We plan

Network-2	Net_O	Net_S
PSNR-T	19.50	24.73
SSIM-T	0.5783	0.7766

Table 2. HDR results for fine-tuning Network-2 using our sensor-realistic synthetic dataset. Both PSNR-T and SSIM-T are improved when tested on real images captured by the modeled sensor. The results are averaged across 15 samples in testset-2

Network-2	S9 Color	S10 Plus Color
PSNR-T	17.95	24.73
SSIM-T	0.6174	0.7766

Table 3. HDR results for fine-tuning Network-2 [52] using the sensor-realistic synthetic data with different color calibration models applied. Network fine-tuned with synthetic data modeled for S10 Plus performs better on captures from S10 Plus. The results are averaged across 15 samples in testset-2.

to conduct a different experiment to train Network-2 from scratch with data from S10 Plus and different forms of synthetic data in future.

Please refer to the supplementary material for more examples showing the experimental results.

6.3. Ablation Study

In addition to the quantitative results shown above, we perform a detailed analysis on Network-2 to study how different sensor models applied to synthetic data can affect the HDR results (Experiment 3).

For analyzing the influence of color calibration, we compare the HDR results between the networks fine-tuned using two versions of sensor-realistic synthetic data with two different color spaces applied (color spaces modeled for a Samsung Galaxy S9 and a S10 Plus accordingly). Table 3 shows that fine-tuning the network using the sensor-realistic synthetic data with the color space modeled for S10 Plus applied yields better HDR results tested on S10 Plus captures.

For analyzing the influence of noise modeling, we compare the HDR results among the networks fine-tuned with multiple versions of the sensor-realistic synthetic dataset with different amount of noise added (ISO levels at 50, 200, and 400). Due to the majority of images in testset-2 being taken at ISO 50 (9 out of 15 sets) and ISO 200 (5 out of 15 sets), with only 1 set of images taken at ISO 320, PSNR value is decreased as more noise (higher ISO) is added to synthetic training data, as shown in Table 4.

Applying the correct sensor realism to the synthetic data is necessary to yield the best HDR result. In other words, data augmentation methods such as randomly adding noise or shifting color will make the network perform even worse. Synthetic data cannot replace true data, at least for HDR applications for now. We will explore training the network with less true data and more sensor-realistic synthetic data in future works.

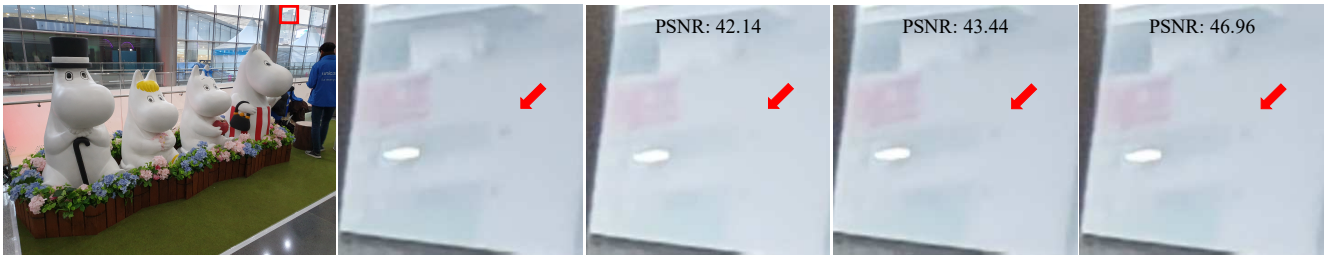


Figure 7. Network-1 fine-tuned with our sensor-realistic synthetic dataset (sensor-realistic) outputs more details (closest to ground-truth) for the building seeing through the window than the outputs from the network fine-tuned with pure synthetic data (pure-synthetic) and the network pre-trained with real data only (pre-trained). Note that the improvement coming from the sensor-realistic synthetic data is not very drastic in Network-1 because this network is already pre-trained with real data captured by the targeted S10 Plus device (the same sensor realism model applied to synthetic data).

Figure 8. Network-2 [52] fine-tuned with our sensor-realistic synthetic dataset (sensor-realistic) outputs more details (window, tree) in the shaded area, while the pre-trained network (pre-trained) fails. Images are equally enhanced by Photomatix [19] for visualization.

Network-2	ISO 50	ISO 200	ISO 400
PSNR-T	24.73	23.81	22.27
PSNR-L	23.67	23.15	22.89

Table 4. HDR results for fine-tuning Network-2 using multiple versions of the sensor-realistic synthetic data with different noise models applied. Testset-2 has more images taken at ISO 50, thus PSNR decreases when more noise (higher ISO) is applied. The results are averaged across 15 samples in testset-2.

7. Conclusion

In this paper, we have presented a sensor-realistic synthetic data generation and processing pipeline to address the data problem in deep learning-based imaging applications on mobile devices. We verify the effectiveness of this pipeline by generating a sensor-realistic synthetic dataset for the data-driven HDR imaging on off-the-shelf smartphone cameras and training two deep HDR networks using this dataset. Experimental results show that both networks yield improved HDR output generated from real images captured by our targeted smartphone cameras when our sensor-realistic synthetic data was used for training the networks. As a starting point, this sensor-realistic synthetic data generation and processing pipeline will be explored more to cover more types of imaging and vision problems with more extensive realism modeled and applied such as scene realism and optical realism.

Acknowledgement We thank all the anonymous reviewers for their valuable comments. We appreciate Jihwan Choe at Visual SW R&D group, Mobile Business, Samsung for the valuable discussion.

References

[1] Miika Aittala and Frédo Durand. Burst image deblurring using permutation invariant convolutional neural networks. In *ECCV*, 2018. 1

[2] A. Atapour-Abarghouei and T. P. Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In *2018 IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, pages 2800–2810, June 2018. 3

[3] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012. 2

[4] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012. 3

[5] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. *CoRR*, abs/1805.01934, 2018. 1

[6] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016. 5

[7] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018. 1

[8] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 369–378, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. 4

[9] Yunbin Deng. Deep learning on mobile devices - A review. *CoRR*, abs/1904.09274, 2019. 1

[10] Hao Dong, Guang Yang, Fangde Liu, Yuanhan Mo, and Yike Guo. Automatic brain tumor detection and segmentation using u-net based fully convolutional networks. In *annual conference on medical image understanding and analysis*, pages 506–517. Springer, 2017. 5

[11] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 2, 3

[12] Gabriel Eilertsen, Joel Kronander, Gyorgy Denes, RafałMantiuk, and Jonas Unger. Hdr image reconstruction from a single exposure using deep cnns. *ACM Transactions on Graphics (TOG)*, 36(6), 2017. 3

[13] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian. Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. *IEEE Transactions on Image Processing*, 17(10):1737–1754, Oct 2008. 4

[14] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. *CoRR*, abs/1605.06457, 2016. 3

[15] Epic Games. UNREAL ENGINE. https://www.unrealengine.com/en-US/, 2019. 2, 3, 4

[16] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. *CoRR*, abs/1604.06646, 2016. 3

[17] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Scenenet: Understanding real world indoor scenes with synthetic data. *CoRR*, abs/1511.07041, 2015. 3

[18] Sam Hasinoff. Introducing the HDR+ Burst Photography Dataset. https://ai.googleblog.com/2018/02/introducing-hdr-burst-photography.html, 2019. 1

[19] HDRsoft. Photomatix. https://www.hdrsoft.com/, 2019. 7, 8

[20] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. *CoRR*, abs/1710.10710, 2017. 3

[21] Yuting Hu, Ruiwen Zhen, and Hamid Sheikh. Cnn-based deghosting in high dynamic range imaging. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4360–4364. IEEE, 2019. 1

[22] Yuan-Ting Hu, Hong-Shuo Chen, Kexin Hui, Jia-Bin Huang, and Alexander G. Schwing. Sail-vos: Semantic amodal instance level video object segmentation - a synthetic dataset and baselines. In *CVPR 2019*, 2019. 3

[23] S. Mohammad Mostafavi I., Lin Wang, Yo-Sung Ho, and Kuk-Jin Yoon. Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks. *CoRR*, abs/1811.08230, 2018. 3

[24] Jk Jensen, Jinhan Hu, Amir Rahmati, and Robert LiKamWa. Protecting visual information in augmented reality from malicious application developers. In *The 5th ACM Workshop on Wearable Systems and Applications*, WearSys '19, pages 23–28, New York, NY, USA, 2019. ACM. 1

[25] Xiaodan Jin and Keigo Hirakawa. Approximations to camera sensor noise. In *IS&T/SPIE Electronic Imaging*, pages 86550H–86550H. International Society for Optics and Photonics, 2013. 4

[26] Nima Khademi Kalantari and Ravi Ramamoorthi. Deep high dynamic range imaging of dynamic scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)*, 36(4), 2017. 2, 3, 4, 6, 7

[27] Graphics Lab. HDR Shop. http://www.hdrshop.com/, 2019. 4

[28] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016. 3

[29] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison. Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2697–2706, Oct 2017. 3

[30] Tom Mertens, Jan Kautz, and Frank Van Reeth. Exposure fusion. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, pages 382–390. IEEE, 2007. 4, 5, 6

[31] Matthias Mueller, Vincent Casser, Jean Lahoud, Neil Smith, and Bernard Ghanem. Ue4sim: A photo-realistic simulator

for computer vision applications. *CoRR*, abs/1708.05869, 2017. 3

[32] Siddhant Prakash, Alireza Bahremand, Linda D. Nguyen, and Robert LiKamWa. Gleam: An illumination estimation framework for real-time photorealistic augmented reality on mobile devices. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '19, pages 142–154, New York, NY, USA, 2019. ACM. 4

[33] Weichao Qiu and Alan L. Yuille. Unrealcv: Connecting computer vision to unreal engine. *CoRR*, abs/1609.01326, 2016. 3

[34] Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Feature mapping for learning fast and accurate 3d pose inference from synthetic images. *CoRR*, abs/1712.03904, 2017. 3

[35] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. *CoRR*, abs/1608.02192, 2016. 3

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2, 5

[37] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243, June 2016. 3

[38] Shunsuke Saito, Lingyu Wei, Liwen Hu, Koki Nagano, and Hao Li. Photorealistic facial texture inference using deep neural networks. *CoRR*, abs/1612.00523, 2016. 3

[39] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser-Nam Lim, and Rama Chellappa. Unsupervised domain adaptation for semantic segmentation with gans. *CoRR*, abs/1711.06969, 2017. 3

[40] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. *CoRR*, abs/1612.07828, 2016. 3

[41] Brian Teixeira, Vivek Singh, Terrence Chen, Kai Ma, Birgi Tamersoy, Yifan Wu, Elena Balashova, and Dorin Comaniciu. Generating synthetic x-ray images of a person from the surface geometry. *CoRR*, abs/1805.00553, 2018. 3

[42] TensorFlow. TensorFlow. https://www.tensorflow.org/, 2019. 6

[43] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. *CoRR*, abs/1804.06516, 2018. 3

[44] Shashank Tripathi, Siddhartha Chandra, Amit Agrawal, Ambrish Tyagi, James M. Rehg, and Visesh Chari. Learning to generate synthetic data via compositing. *CoRR*, abs/1904.05475, 2019. 3

[45] Apostolia Tsirikoglou, Joel Kronander, Magnus Wrenninge, and Jonas Unger. Procedural modeling and physically based rendering for synthetic data generation in automotive applications. *CoRR*, abs/1710.06270, 2017. 3

[46] Okan Tarhan Tursun, Ahmet Oguz Akyüz, Aykut Erdem, and Erkut Erdem. The state of the art in hdr deghosting: A survey and evaluation. *Comput. Graph. Forum*, 34:683–707, 2015. 1

[47] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. *CoRR*, abs/1701.01370, 2017. 3

[48] Neal Wadhwa, Rahul Garg, David E. Jacobs, Bryan E. Feldman, Nori Kanazawa, Robert Carroll, Yair Movshovitz-Attias, Jonathan T. Barron, Yael Pritch, and Marc Levoy. Synthetic depth-of-field with a single-camera mobile phone. *ACM Trans. Graph.*, 37(4):64:1–64:13, July 2018. 1

[49] Qi Wang, Junyu Gao, Wei Lin, and Yuan Yuan. Learning from synthetic data for crowd counting in the wild. *CoRR*, abs/1903.03303, 2019. 3

[50] Wikipedia. Gamut. https://en.wikipedia.org/wiki/Gamut, 2019. 4

[51] Wikipedia. High-dynamic-range imaging. https://en.wikipedia.org/wiki/High-dynamic-range_imaging, 2019. 2

[52] Shangzhe Wu, Jiarui Xu, Yu-Wing Tai, and Chi-Keung Tang. End-to-end deep HDR imaging with large foreground motions. *CoRR*, abs/1711.08937, 2017. 2, 3, 4, 5, 6, 7, 8

[53] x rite. ColorChecker Classic. https://xritephoto.com/colorchecker-classic, 2019. 2

[54] Mengwei Xu, Jiawei Liu, Yuanqiang Liu, Felix Xiaozhu Lin, Yunxin Liu, and Xuanzhe Liu. When mobile apps going deep: An empirical study of mobile deep learning. *CoRR*, abs/1812.05448, 2018. 1

[55] Qingsen Yan, Dong Gong, Qinfeng Shi, Anton van den Hengel, Chunhua Shen, Ian D. Reid, and Yanning Zhang. Attention-guided network for ghost-free high dynamic range imaging. *CoRR*, abs/1904.10293, 2019. 3

[56] Xin Yang, Ke Xu, Yibing Song, Qiang Zhang, Xiaopeng Wei, and Rynson W. H. Lau. Image correction via deep reciprocating HDR transformation. *CoRR*, abs/1804.04371, 2018. 3

[57] Jiachao Zhang, Keigo Hirakawa, and Xiaodan Jin. Quantile analysis of image sensor noise distribution. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015. 4

[58] Jinsong Zhang and Jean-François Lalonde. Learning high dynamic range from outdoor panoramas. *CoRR*, abs/1703.10200, 2017. 3

[59] Yi Zhang, Weichao Qiu, Qi Chen, Xiaolin Hu, and Alan L. Yuille. Unrealstereo: A synthetic dataset for analyzing stereo vision. *CoRR*, abs/1612.04647, 2016. 3