Supplementary Material
Graphics2RAW: Mapping Computer Graphics Images to Sensor RAW Images

Donghwan Seo[*1] Abhijith Punnappurath[*2] Luxi Zhao[2] Abdelrahman Abdelhamed[†3]
Sai Kiran Tedla[‡2,4] Sanguk Park[1] Jihwan Choe[1] Michael S. Brown[2]
[1]Samsung Electronics [2]Samsung AI Center Toronto [3]Google Research [4]York University, Toronto

This supplementary material contains additional results and experimental details. In Section S1, we discuss the implementation details of the three tasks performed in the main paper – RAW denoising, illuminant estimation, and neural rendering. Representative examples from our nighttime CG dataset as well as additional visual results, are provided in Section S2. Finally, Section S3 examines the challenging scenario where just a *single* DNG from the target sensor is available, as against a set of DNGs.

S1. Implementation details

Our method assumes that a small set of RAW files from the target sensor is available in DNG format. The Digital Negative (DNG) format [1] is a popular open RAW file format introduced by Adobe. While most modern cameras natively support DNG, nearly all RAW formats can be converted to DNG using Adobe's DNG converter. Therefore, there is no loss of generality in our choice of RAW DNGs. The real RAW file in Fig. 1 of our main paper is from the Nikon D40 DSLR camera from the NUS dataset [4]. The RAW image is originally in Nikon's NEF file format and has been converted to DNG using Adobe's DNG converter. Our graphics RAW image in Fig. 1 of our main paper used the Samsung S20 FE smartphone, which natively supports RAW DNG, as our target sensor. Note that the sRGB rendering of our graphics DNG and the real RAW image were obtained using Photoshop's Auto White Balance (AWB) and Auto Adjustments settings.

As stated in the main paper in Section 1, graphics renderings are implicitly white-balanced. Since the color of lighting is specified as part of the rendering, any light source that deviates from white (e.g., a blue light) can be treated as a desired aesthetic (i.e., a 3D graphics artist would not change a light source to blue only to have it white-balanced in rendering). Some graphics engines, such as Unity, have optional post-processing tools for color grading. For example, it is possible to adjust the overall color cast/tint and brightness levels after rendering. However, Unity's post-rendering color manipulation is intended to match rendered graphics images to appear visually similar to other imagery (camera captured or other graphics renderings). We are not aware of any graphics-based synthetic data used for computer vision that includes additional color grading steps after rendering. As a result, we assume that the graphics sRGB images used by our method do not include additional post-processing color manipulation.

We begin with details regarding network architectures, datasets, and training parameters for all three tasks discussed in the main paper—(1) RAW denoising, (2) illuminant estimation, and (3) neural ISP. We want to emphasize that achieving state-of-the-art results on these specific tasks was *not* the objective of these experiments. Each topic is an active research area, and many DNN-based methods exist in the literature. Instead, the goal was to evaluate the quality of our *graphics to RAW* compared to alternative strategies for different sensors and tasks. Specifically, for each task, we chose a representative network architecture and trained it using (1) our proposed approach; (2) UPI [3]; (3) EnlightenGAN [7]; and (4) real RAW image data from the target sensors. We will describe each of these tasks in more detail in the following.

For all our experiments in the main paper, the graphics data was encoded in a gamma-sRGB color space which is part of the sRGB standard. In the case of the SYNTHIA dataset [9], the graphics images were rendered with a gamma applied. For our nighttime CG dataset generated using the Unreal Engine 5 [2], the images are saved as linear-sRGB without the gamma (an option in Unreal). We applied a 2.2 gamma to these images for the sake of uniformity of the data across experiments. Also note that UPI [3] and EnlightenGAN [7] are designed for gamma sRGB images, and applying the gamma to our dataset ensures a fair comparison against these methods. When applying our method, we undo the gamma as the very first step of our pipeline (al-

[*]Equal contribution.
[†]Work done while with the Samsung AI Center Toronto.
[‡]Work done while an intern at the Samsung AI Center Toronto.

though this is not explicitly shown in Fig. 3 of our main paper).

We also note that we compared our method to an *unpaired* generative method, namely EnlightenGAN [7], because *supervised* DNN methods, such as Cycle ISP [14] and Invertible ISP [11], require paired RAW-sRGB images for training. As discussed in the main paper, these methods cannot be applied to graphics data since no ground truth RAW images correspond to the graphics images.

S1.1. RAW denoising

For the RAW denoising experiment, we used the real noisy RAW images from the nighttime dataset of [8]. There are 105 noisy images at ISO 1600 and 3200. All scenes were static and captured using a tripod. The ground truth RAW images in [8] were computed by averaging 30 frames at ISO 50. In Table 1 of the main paper, the graphics-based models in the first three rows were tested on these 105 images. The real data model in the last row was trained and evaluated on this same data using three-fold cross-validation, with the training, validation, and testing split indices defined in [8].

The Restormer architecture from [13] was used as the denoiser. We used the official implementation from the authors. The original Restormer model was designed for 3-channel sRGB images. Since we are targeting RAW denoising, we modified the architecture to accept, and output, 4-channel stacked RGGB Bayer RAW images. All models were trained for 100 epochs.

All graphics-based models used 70 images randomly sampled from our CG nighttime dataset – 60 for training and 10 for validation. This division was done to be consistent with the models trained on real RAW images that also received 60/10 images for training/validation during three-fold cross-validation. Since all images (real and synthetic) were high-resolution ($\approx 3000 \times 4000$ pixels), we found we had sufficient patches for training the Restormer. As mentioned in the main paper, the noise generator from [8] was used to add synthetic noise to all models trained using graphics-based (i.e., synthetic) RAW images. The parameters of the heteroscedastic Gaussian noise model used by [8] were estimated based on a calibration procedure. The UPI method [3] also includes a heteroscedastic Gaussian noise model that is fit using the read and shot noise parameters in the RAW metadata. However, we found the results of the *calibrated* heteroscedastic Gaussian noise model of [8] to be more accurate. Therefore, for a fair comparison, we used this noise model for all methods, including UPI.

Finally, we would like to note that while our method can easily adopt any noise model, our focus in this work is not on proposing a state-of-the-art RAW noise generator. Depending on the requirements of the task, noise can be added to our graphics RAW image using any existing noise model.

Table S1. Illuminant estimation results using the C4 [12] network on the NUS dataset [4]. Angular errors are reported.

Model	Mean	Median	Top 25%	Worst 25%
UPI [3]	3.39	2.69	0.85	7.06
Ours	**2.79**	**2.12**	**0.70**	**5.97**
Real	2.45	1.69	0.52	5.62

Adding noise is an optional step in our processing pipeline, and is therefore omitted from Fig. 3 of our main paper. For the following two tasks of illuminant estimation and neural ISP, we do not add noise.

S1.2. Illuminant estimation

The illumination estimation task was evaluated on the NUS color constancy dataset [4]. The ground truth illuminations are available as part of the dataset through a Macbeth color checker chart placed in each scene. To train the real data models, we randomly selected 125 images for training and 25 for validation for each of the nine cameras in the dataset. The remaining images were held out for testing. The color chart was masked out both during training and testing.

We used the CNN architecture from [6] as our illuminant estimation network. Since the author of [6] has yet to release the code, we used our implementation. We used the same loss function, learning rate, and other training parameters described in the paper. In [6], the model was trained by cropping patches from the training images and downsampled thumbnail versions of the same images in the training set were used for validation. In our experiments, we used patches cropped from a separate validation set to select the best model.

For the illuminant estimation experiment, our method, as well as EnlightenGAN [7] and UPI [3], were applied to graphics images from the SYNTHIA dataset [9]. We randomly sampled 125 images from this dataset for training and 25 for validation. The images in the SYNTHIA dataset have a resolution of 720×960 pixels, so we downsampled the images from the NUS dataset by a factor of four so that the image resolutions are roughly the same. Visualizations of the synthetic RAW datasets generated by these methods, as well as real RAW data corresponding to two cameras from the NUS [4] dataset, are provided in Figs. S1 and S2. It can be clearly observed that our method's color distribution is closer to the real data compared to EnlightenGAN and UPI. As evident from the quantitative results in Table 2 of our main paper, our method also performs significantly better than our closest competitor UPI. Fig. S3 shows some qualitative comparisons.

The illumination estimation network in [6] is a lightweight model containing only around 1K parameters. For comparison, we repeated the illuminant estimation ex-

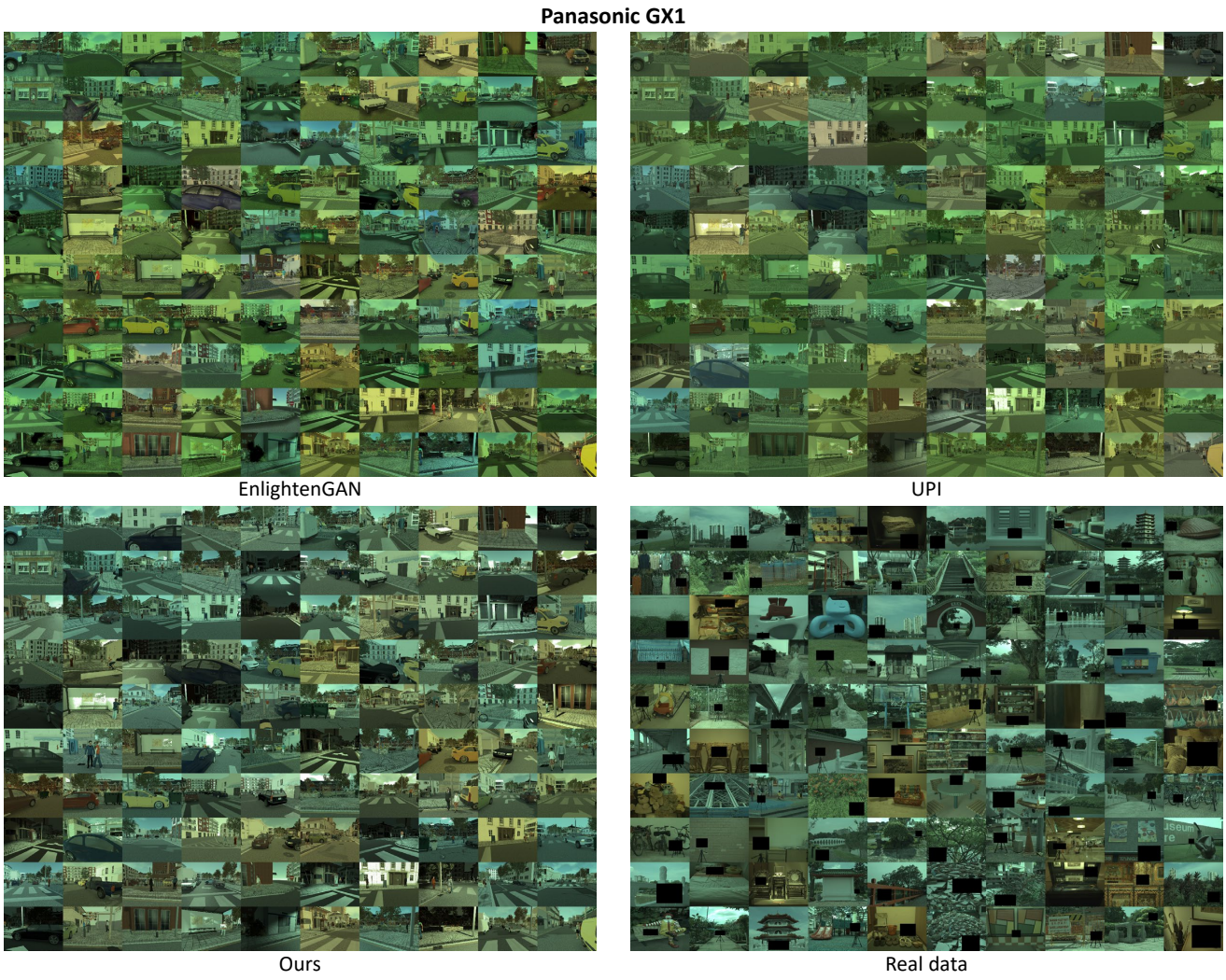


Figure S1. Samples from the synthetic RAW datasets generated by EnlightenGAN [7], UPI [3], and our method, and real RAW data from the Panasonic GX1 camera of the NUS dataset [4].

periments of Section 4.4 of the main paper with this network replaced with the more complex model of C4 [12]. Results are reported in Table S1. Since C4 is a much larger capacity model, the errors drop for all methods compared to Table 2 of our main paper. However, the same trend can be observed with our method outperforming our closest competitor UPI and having only a small gap to the models trained on real data.

S1.3. Neural ISP

We used the nighttime dataset of [8] to examine our method's performance on the neural rendering task. We selected the 105 clean RAW images from the dataset for evaluation. We used the official code released by the authors for training and testing. As mentioned in the paper, we used Photoshop to render the RAW images to sRGB instead of using their simplified software ISP code. We did observe that this led to slightly different PSNR (dB) and SSIM scores than reported in their paper [8]. Similar to the denoising task, the real data models were trained and evaluated using three-fold cross-validation with the dataset partitions defined in [8]. The real models used 60/10 images for training/validation. Therefore, we randomly sampled 60/10 images from our nighttime dataset to train/validate the three graphics-based methods—EnligtenGAN, UPI, and our method.

Qualitative comparisons are provided in Fig. S4. Additionally, we compared against the day-to-night approach proposed in [8], and quantitative results are presented in Table S2. The results of other methods are reproduced from the main paper for ease of comparison. We outperform competitors by a sound margin and perform on par with the models trained on real data. As an additional comparison, we replaced the UNet architecture used as the ISP

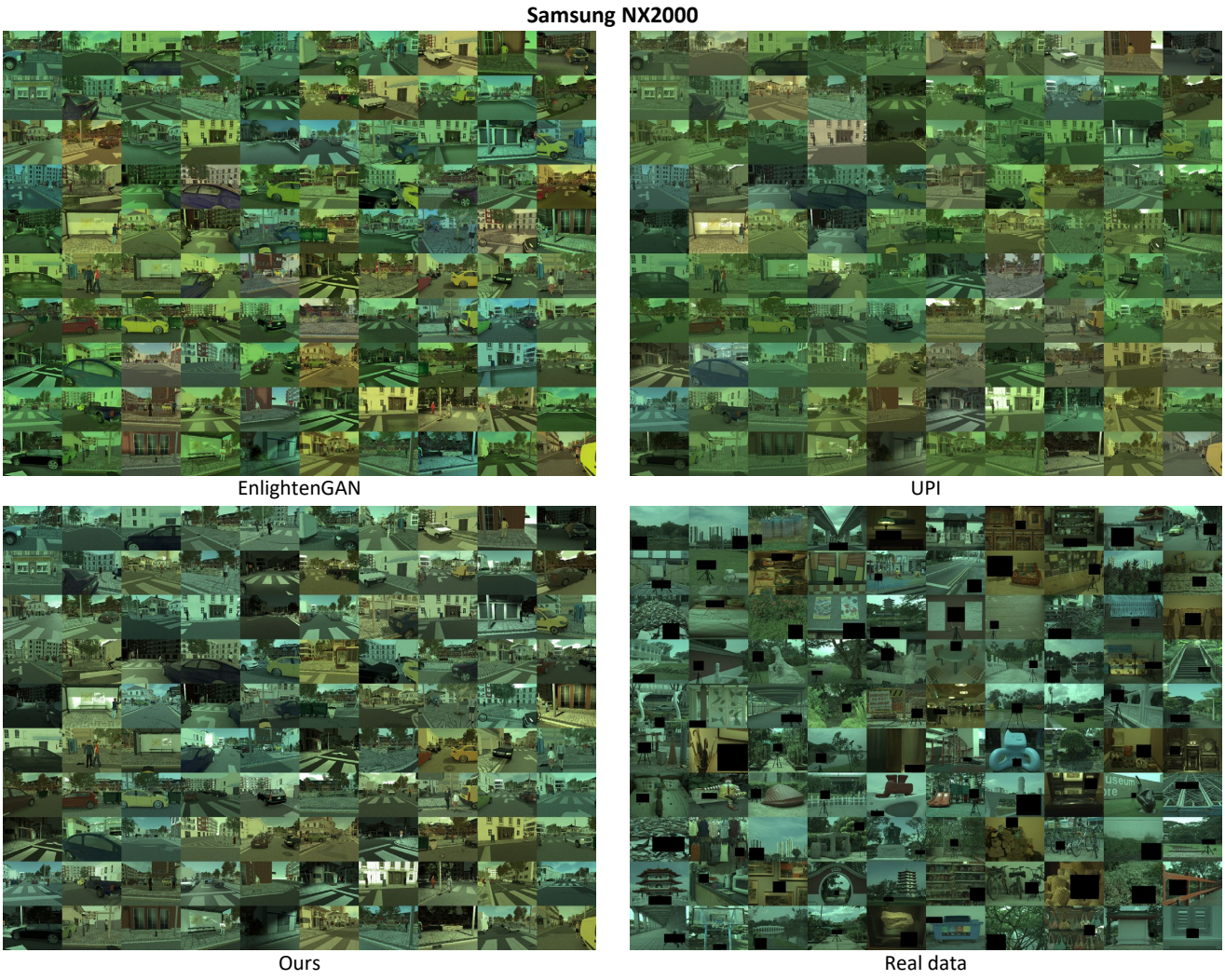


Figure S2. Samples from the synthetic RAW datasets generated by EnlightenGAN [7], UPI [3], and our method, and real RAW data from the Samsung NX2000 camera of the NUS dataset [4].

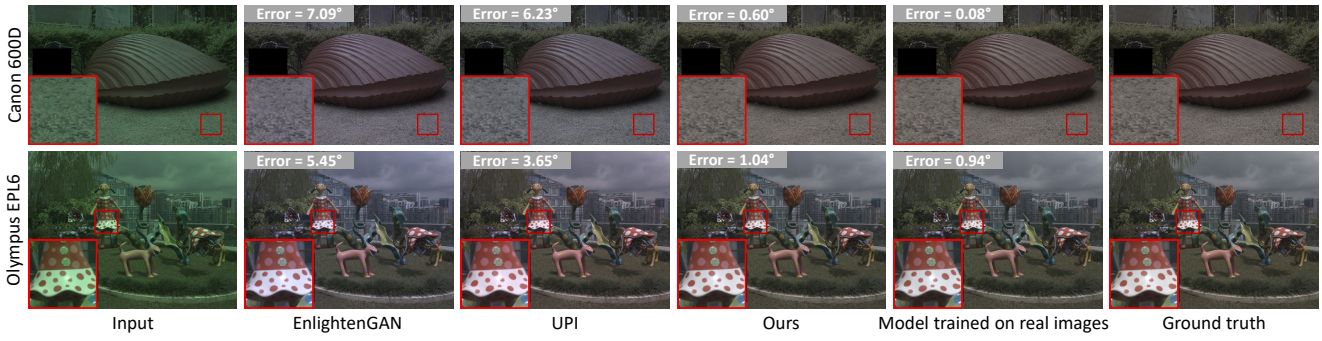


Figure S3. Qualitative results for our illuminant estimation task. Inset shows angular error value.

in [8] with MW-ISPNet [5]—winner of the AIM Learned ISP Challenge at ECCV'20. Results are reported in Table S3. Even with a more sophisticated network, similar trends can be observed with our method producing a convincing 1 dB improvement over UPI.

Visualizations of the synthetic RAW datasets generated by day to night [8], EnlightenGAN [7], UPI [3], and our method, as well as real RAW data from the Samsung S20 FE

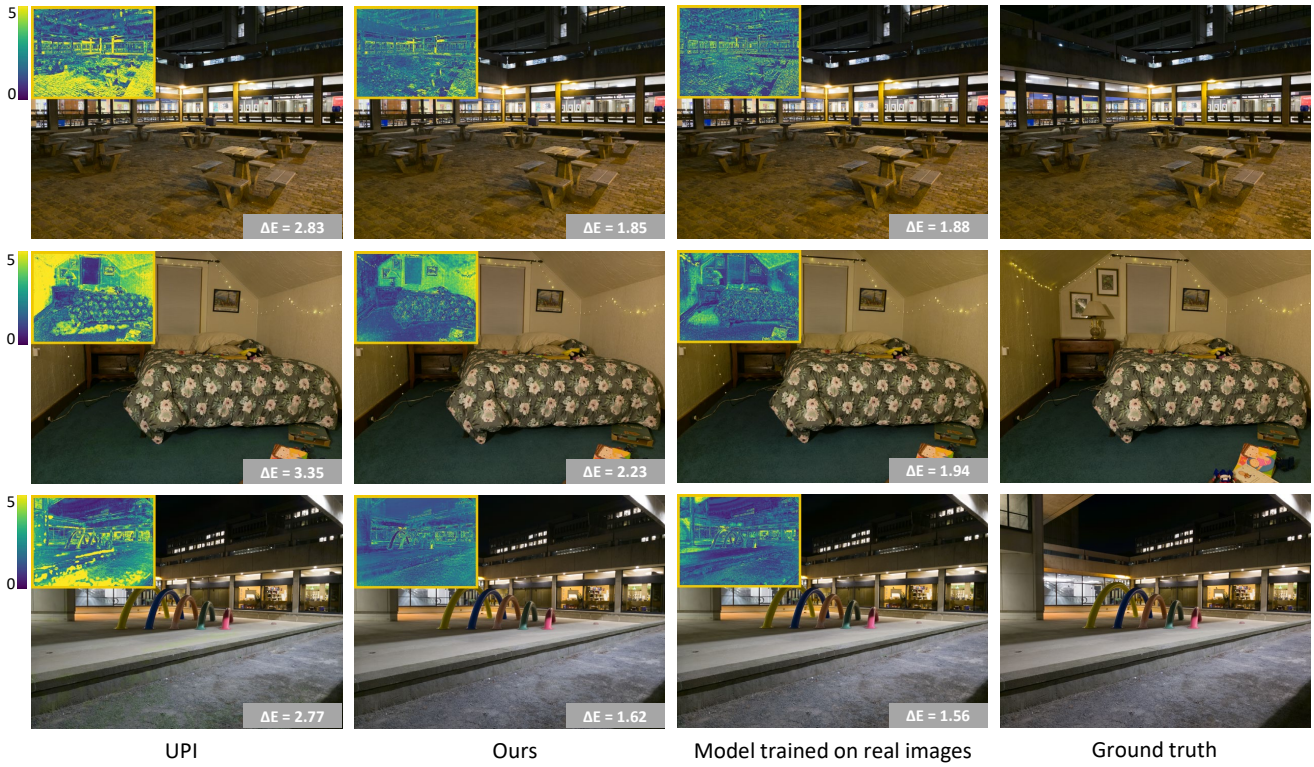


Figure S4. Qualitative results for our neural ISP task. Inset shows ΔE [10] error map and average value.

Table S2. Quantitative results on our neural ISP task on the nighttime dataset of [8].

Model	PSNR ↑	SSIM ↑	ΔE ↓ [10]
Day to night [8]	35.64	0.973	2.695
EnlightenGAN [7]	35.58	0.965	3.137
UPI [3]	36.43	0.966	2.907
Ours	**38.10**	**0.974**	**2.301**
Real	38.32	0.974	2.133

Table S3. Quantitative results on our neural ISP task on the nighttime dataset of [8] using MW-ISPNet [5].

Model	PSNR ↑	SSIM ↑	ΔE ↓ [10]
UPI [3]	37.80	0.970	2.267
Ours	**38.95**	**0.972**	**2.014**
Real	40.12	0.977	1.747

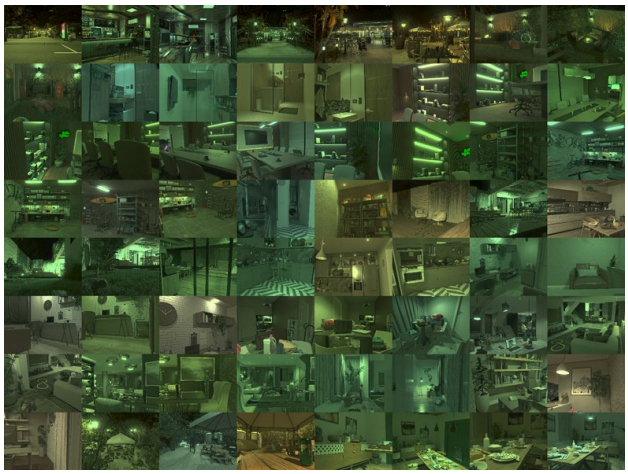
smartphone camera, are provided in Fig. S5. Once again, it can be clearly observed that our method's color distribution is closer to the real data compared to all other methods.

S2. Additional qualitative results

We show representative samples from our nighttime graphics dataset in Fig. S6. The images are generated using Unreal Engine 5 [2] and saved as 32-bit linear sRGB images in EXR format with a resolution of 3000×4000 pixels. For visualization, a 2.2 gamma has been applied to the images in Fig. S6. Note that the Unreal graphics engine has settings to model camera lens-specific effects (e.g., vignetting and chromatic aberration). However, including such effects in the rendering would limit the ability to reuse our CG dataset for multiple sensors. Instead, we assume that real RAW test images are preprocessed using DNG metadata to remove such degradations e.g., applying lens shading correction to remove chromatic aberration and vignetting, and so we generate CG images without these effects.

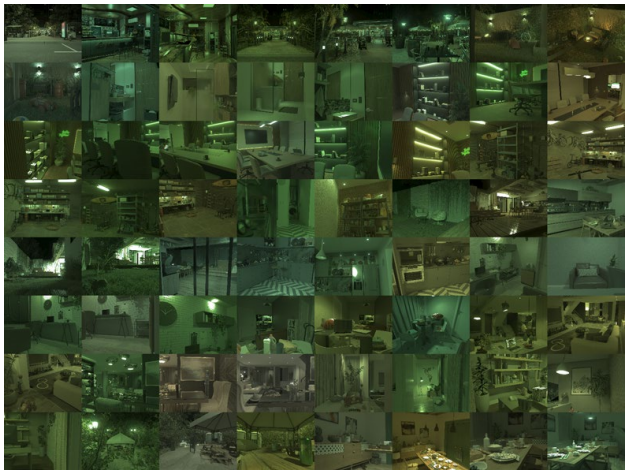
Figs. S7, S8, and S9 provide additional examples similar to Fig. 5 in the main paper and show qualitative comparisons between EnlightenGAN [7], UPI [3], and our method. The figures show synthetic RAW images generated by each method rendered back to sRGB. Note that our objective is not to convert the graphics image to a RAW image such that when the synthetic RAW image is rendered back to sRGB, it matches exactly the CG image. Instead, we can think of the CG image as a proxy for a physical scene radiance image that has been rendered to an sRGB image by an ideal camera. Our graphics-to-RAW method aims to map this ideal CG sRGB image to a sensor-specific RAW image under a particular illumination. When the synthesized RAW image is rendered, the photofinishing routines of the target camera will be applied to impart the look and feel of that camera. The parameters of these photofinishing steps are

Real data



EnlightenGAN

Day to night



UPI

Ours

Figure S5. Samples from the synthetic RAW datasets generated by EnlightenGAN [7], day to night [8], UPI [3], and our method, and real RAW data from the Samsung S20 FE smartphone camera from the nighttime dataset of [8].

Figure S6. Representative examples from our nighttime graphics dataset.

often embedded in the camera's DNG files. Software like Adobe Photoshop will emulate the camera's ISP by applying these steps using the DNG metadata. As a result, there will be subtle differences among different cameras. However, we do expect the camera-rendered images to look like the output that the target camera would produce. An improper sampling of the sensor's CST and illumination leads to poor results, as the RAW image values do not reflect plausible RAW RGB values that match the DNG—rendering an incorrectly synthesized RAW image results in noticeable color artifacts and color casts in the rendered images, as observed in the outputs of EnlightenGAN and UPI.

Figs. S7, S8, and S9 show two different scenes for each camera that are synthesized under different illuminations. For the first example corresponding to each scene, the illuminant is chosen to be the same for EnlightenGAN [7], UPI [3], and our method. This illuminant sample lies close to the distribution of the ground truth illuminants (see the chromaticity plots in the figure). UPI's output has a noticeable color cast even when the illuminant is sampled close to the ground truth illuminants. EnlightenGAN produces undesirable color artifacts – green-colored regions on the road, buildings, etc. For the second example under each scene, different randomly-sampled illuminants are used for the three methods. Our method produces more realistic color renderings compared to other approaches.

S3. Using a single DNG

For all our experiments in the main paper, we used a *set* of DNGs to construct the distribution of illuminants. Here, we analyze the limiting case when just a *single* DNG from the target sensor is available. Given a DNG from the target sensor, we first map the two illuminants corresponding to the two pre-calibrated CST matrices stored in the DNG metadata to the sensor's color space. In particular, since the CCT values of the two illuminants can be determined from the 'CalibrationIlluminant1' and 'CalibrationIlluminant2' tags, their corresponding CIE XYZ values can be computed. Applying the inverse of each CST matrix to the corresponding estimated CIE XYZ value gives us the location of the two illuminants in the sensor's color space. This is indicated by the red and green circles in the chromaticity plot of Fig. S10(a). The plot corresponds to the Sony A57 camera from the NUS [4] dataset, which uses D65 and Standard Light A as the calibration illuminants. We can also generate samples between these two extreme color temperature values by interpolating between the CSTs. This produces the CCT locus indicated by the small pink circles. Finally, we can build a distribution around these samples using equations (2) and (3) of our main paper and randomly sample illuminant values from this distribution, as shown in the plot.

Fig. S10(b) shows a comparison between the ground

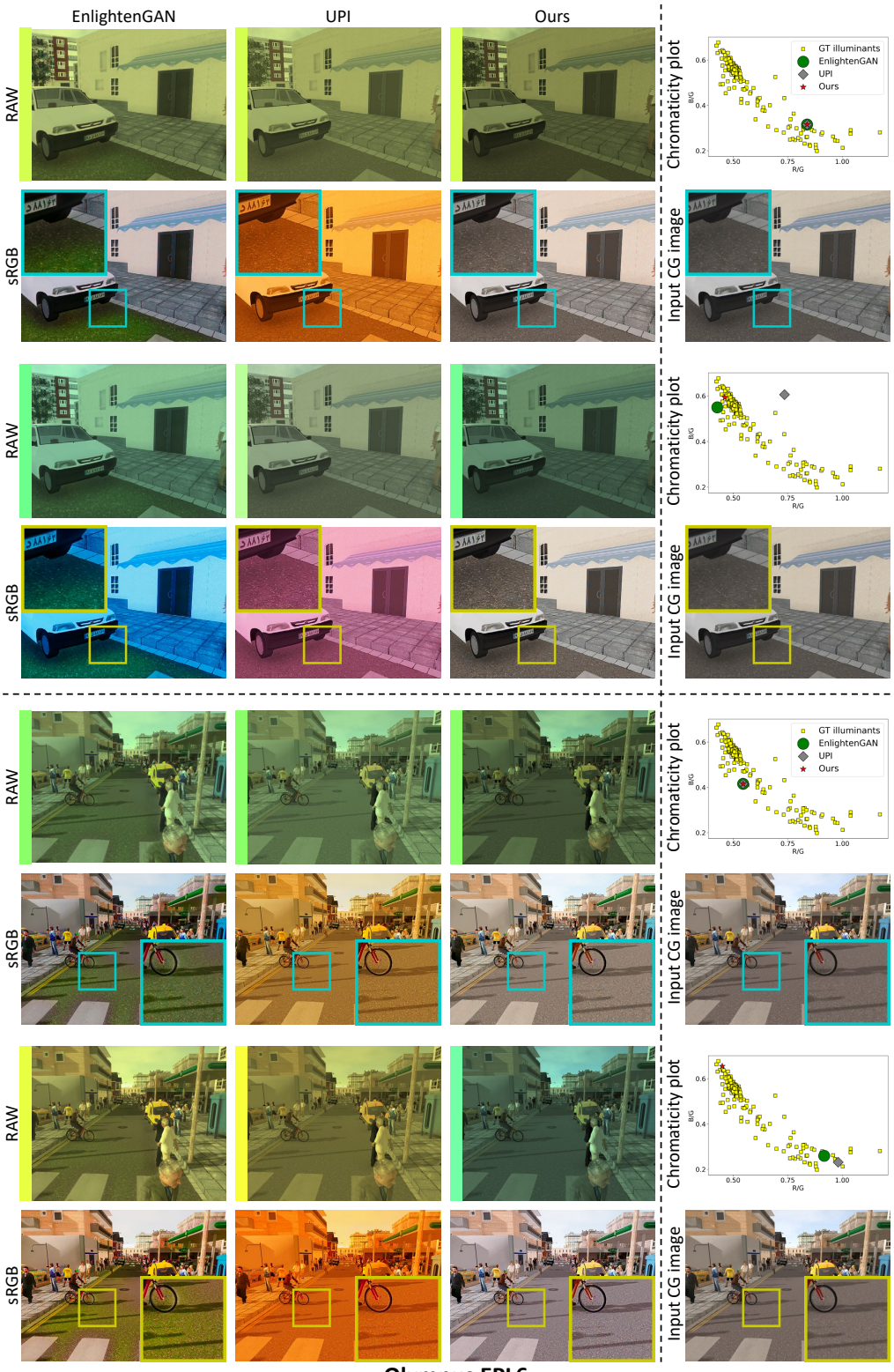


Figure S7. Qualitative comparisons between EnlightenGAN [7], UPI [3], and our method. The Olympus EPL6 DSLR camera from the NUS dataset [4] is used as the target sensor. The images are from the SYNTHIA dataset [9]. Vertical bars represent the color of the illuminant in the RAW space. Two scenes are shown under different illuminations. For each scene, the illuminant in the first example is chosen to be the same for EnlightenGAN [7], UPI [3], and our method, while the illuminants are different across methods in the second example, as shown by the chromaticity plots. The sRGB outputs are rendered using Photoshop under the AWB setting. Insets show zoomed-in regions.

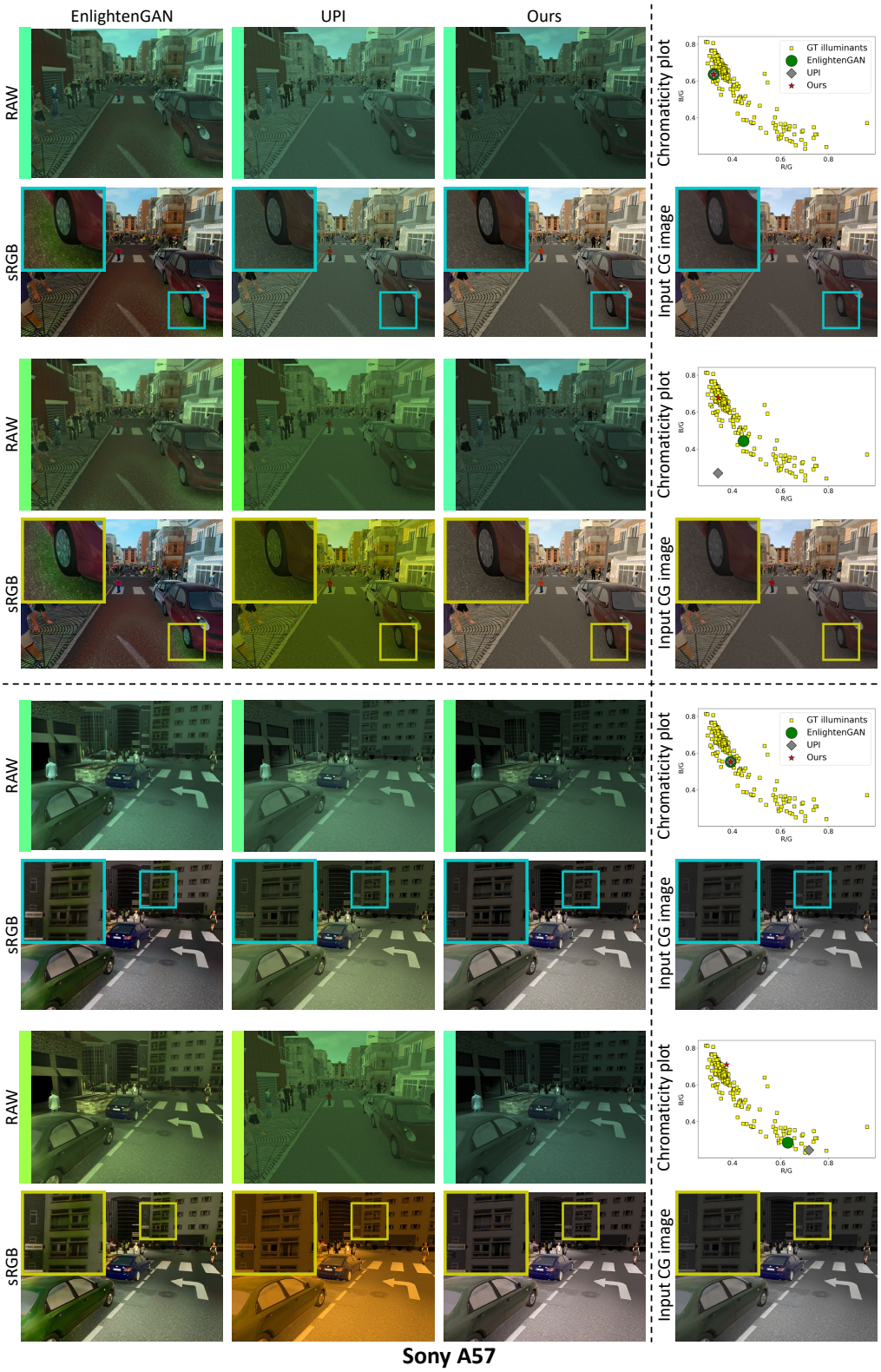


Figure S8. Qualitative comparisons between EnlightenGAN [7], UPI [3], and our method. The Sony A57 DSLR camera from the NUS dataset [4] is used as the target sensor. The images are from the SYNTHIA dataset [9]. Vertical bars represent the color of the illuminant in the RAW space. Two scenes are shown under different illuminations. For each scene, the illuminant in the first example is chosen to be the same for EnlightenGAN [7], UPI [3], and our method, while the illuminants are different across methods in the second example, as shown by the chromaticity plots. The sRGB outputs are rendered using Photoshop under the AWB setting. Insets show zoomed-in regions.

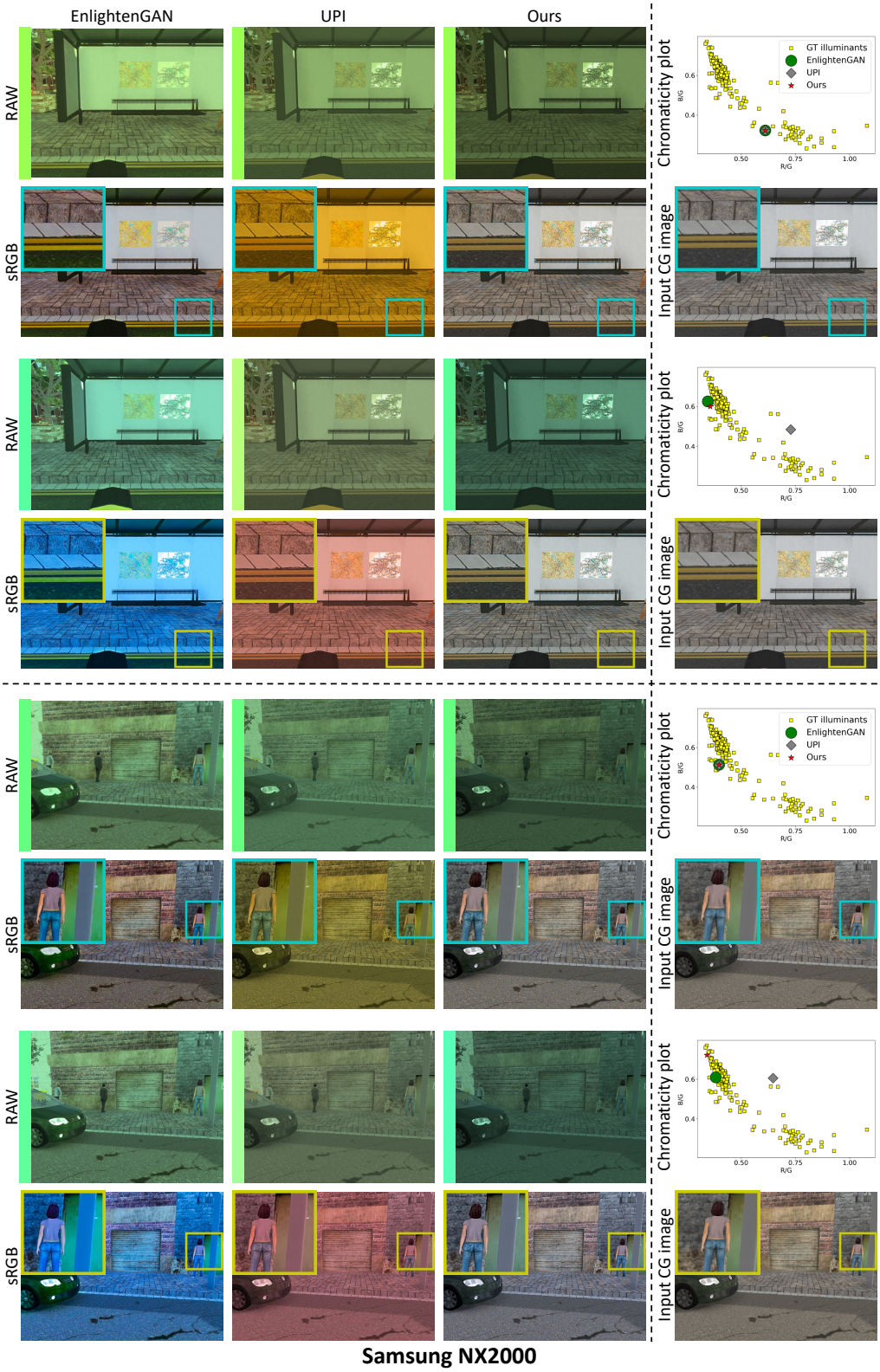
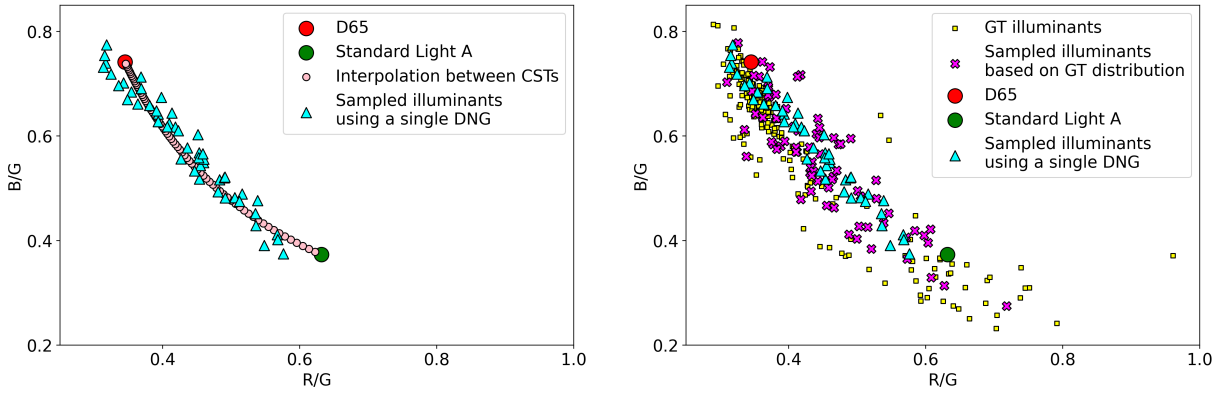
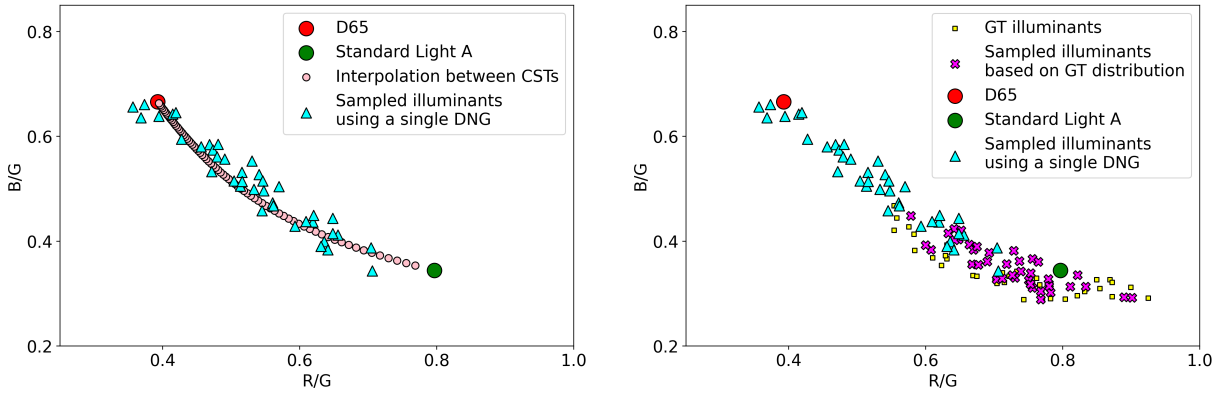


Figure S9. Qualitative comparisons between EnlightenGAN [7], UPI [3], and our method. The Samsung NX2000 DSLR camera from the NUS dataset [4] is used as the target sensor. The images are from the SYNTHIA dataset [9]. Vertical bars represent the color of the illuminant in the RAW space. Two scenes are shown under different illuminations. For each scene, the illuminant in the first example is chosen to be the same for EnlightenGAN [7], UPI [3], and our method, while the illuminants are different across methods in the second example, as shown by the chromaticity plots. The sRGB outputs are rendered using Photoshop under the AWB setting. Insets show zoomed-in regions.



(a) (b)

Figure S10. Sampling the illuminant space when only a single RAW DNG from the target sensor is available. The chromaticity plots show the sensor color space for the Sony A57 DSLR camera from the NUS dataset [4].



(a) (b)

Figure S11. Sampling the illuminant space when only a single RAW DNG from the target sensor is available. The chromaticity plots show the sensor color space for the Samsung S20 FE smartphone camera from the nighttime dataset of [8].

truth illuminants, random samples drawn from a distribution constructed using the ground truth illuminant values, and random illuminants obtained by sampling around the CCT locus when only a single DNG is available. It can be seen from the plot that even with a single DNG, we can well approximate the distribution of real illuminations.

To analyze the accuracy of our single-DNG sampling strategy, we perform the illuminant estimation task of Section 4.4 of our main paper. The results are reported in Table S4. For ease of comparison, the results of other methods are reproduced from Table 2 of the main paper. Our single DNG approach, while less accurate than our proposed framework, still outperforms UPI and EnlightenGAN.

We also test our single DNG technique on the nighttime neural ISP task of Section 4.5 of our main paper. The chromaticity plots of Fig. S11 show the illuminant sampling for the S20 FE camera from the nighttime dataset of [8]. Here, it can be observed that the illuminant samples from a single DNG only partially overlap with the ground truth illuminants – many nighttime illuminant samples lie well beyond Standard Light A, and there are very few nighttime illuminants close to D65. The results of the single DNG approach are compared with our proposed method's results in Table S5. The single DNG method does not perform well due to the gap from real nighttime illuminations.

Table S4. Illuminant estimation results on the NUS dataset [4]. Angular errors are reported.

Model	Mean	Median	Top 25%	Worst 25%
EnlightenGAN [7]	7.01	6.82	3.48	11.07
UPI [3]	6.26	5.89	2.92	10.33
Ours single DNG	4.59	3.41	1.34	10.11
Ours	4.21	3.38	1.30	8.57
Real	3.02	2.17	0.75	6.77

We also conduct an ablation by varying the number of DNGs for the tasks of illumination estimation and neural ISP. While performance may be subpar on the neural ISP task with a single DNG, results significantly improve with

Table S5. Quantitative comparison between the single DNG approach and our proposed framework for our neural ISP task on the nighttime dataset of [8].

Model	PSNR ↑	SSIM ↑	ΔE ↓ [10]
Ours single DNG	35.74	0.963	3.063
Ours	38.10	0.974	2.301

Table S6. An ablation on the number of DNGs evaluated on the tasks of illuminant estimation and neural ISP.

#DNGs Illum. estimation	1	25	75	150
Mean angular error ↓	4.59	4.50	4.40	4.21
#DNGs Neural ISP	1	10	20	40
PSNR (dB) ↑	35.74	37.87	37.84	38.10

as few as 10 or 25 DNGs.

This final experiment demonstrates the importance of real data samples for certain applications. This experiment also reinforces our argument that accurate illuminant sampling is critical for precise color reproduction tasks. We applied a straightforward multivariate Gaussian function in our framework to model the illuminant distribution. In future work, we plan to explore more accurate task-specific illuminant sampling methods given additional knowledge of the target sensor, such as the sensor's spectral sensitivity profile.

References

[1] Adobe Digital Negative (DNG). https://helpx.adobe.com/camera-raw/digital-negative.html. Accessed: 2022-11-06. 1

[2] Unreal Engine. https://www.unrealengine.com/unreal-engine-5. Accessed: 2022-11-06. 1, 5

[3] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T. Barron. Unprocessing images for learned raw denoising. In *CVPR*, 2019. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

[4] Dongliang Cheng, Dilip K. Prasad, and Michael S. Brown. Illuminant estimation for color constancy: Why spatial-domain methods work and the role of the color distribution. *Journal of the Optical Society of America A*, 31(5):1049–1058, 2014. 1, 2, 3, 4, 7, 8, 9, 10, 11

[5] Ignatov et al. AIM 2020 challenge on learned image signal processing pipeline. In *ECCV Workshop*, 2020. 4, 5

[6] Han Gong. Convolutional mean: A simple convolutional neural network for illuminant estimation. In *BMVC*, 2019. 2

[7] Yifan Jiang, Xinyu Gong, Ding Liu, Yu Cheng, Chen Fang, Xiaohui Shen, Jianchao Yang, Pan Zhou, and Zhangyang Wang. EnlightenGAN: Deep light enhancement without paired supervision. *TIP*, 30:2340–2349, 2021. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

[8] Abhijith Punnappurath, Abdullah Abuolaim, Abdelrahman Abdelhamed, Alex Levinshtein, and Michael S. Brown. Day-to-night image synthesis for training nighttime neural ISPs. In *CVPR*, 2022. 2, 3, 4, 5, 6, 11, 12

[9] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 1, 2, 8, 9, 10

[10] Gaurav Sharma and Raja Bala. *Digital Color Imaging Handbook*. CRC Press, 2nd edition, 2013. 5, 12

[11] Yazhou Xing, Zian Qian, and Qifeng Chen. Invertible image signal processing. In *CVPR*, 2021. 2

[12] Huanglin Yu, Ke Chen, Kaiqi Wang, Yanlin Qian, Zhaoxiang Zhang, and Kui Jia. Cascading convolutional color constancy. In *AAAI*, 2020. 2, 3

[13] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *CVPR*, 2022. 2

[14] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. CycleISP: Real image restoration via improved data synthesis. In *CVPR*, 2020. 2