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

Abstract

Computer graphics (CG) rendering platforms produce imagery with ever-increasing photo realism. The narrowing domain gap between real and synthetic imagery makes it possible to use CG images as training data for deep learning models targeting high-level computer vision tasks, such as autonomous driving and semantic segmentation. CG images, however, are currently not suitable for low-level vision tasks targeting RAW sensor images. This is because RAW images are encoded in sensor-specific color spaces and incur pre-white-balance color casts caused by the sensor's response to scene illumination. CG images are rendered directly to a device-independent perceptual color space without needing white balancing. As a result, it is necessary to apply a mapping procedure to close the domain gap between graphics and RAW images. To this end, we introduce a framework to process graphics images to mimic RAW sensor images accurately. Our approach allows a one-to-many mapping, where a single graphics image can be transformed to match multiple sensors and multiple scene illuminations. In addition, our approach requires only a handful of example RAW-DNG files from the target sensor as parameters for the mapping process. We compare our method to alternative strategies and show that our approach produces more realistic RAW images and provides better results on three low-level vision tasks: RAW denoising, illumination estimation, and neural rendering for night photography. Finally, as part of this work, we provide a dataset of 292 realistic CG images for training low-light imaging models.

1. Introduction

Access to large datasets of high-quality images remains a critical factor for training deep learning models. For many computer vision tasks, data acquisition is time-consuming, expensive, and error prone. This is particularly true for camera engineers who often require sensor-specific training data. Any time a new sensor is introduced on a smartphone or other camera platform, training images specific to the new sensor must be captured. Computer-generated (CG) imagery has long been viewed as a promising solution to the data acquisition bottleneck. The main challenge with graphics imagery is the domain gap between synthetic and real data. Fortunately, the visual gap is rapidly closing thanks to the improved realism of graphics engines.

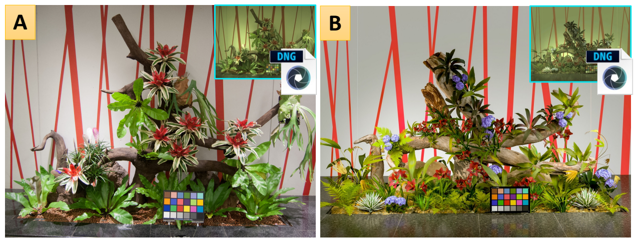

Figure 1. Images (A) and (B) started as RAW-DNG images, one from a Nikon, the other from a Samsung camera. RAW images are shown as insets. Both DNGs have been rendered by Adobe Photoshop to sRGB based on the DNG's metadata. One is a computer graphics image converted to a RAW-DNG using our method; the other is a real sensor RAW image. Can you tell which is CG and which is real? See Sec. 3.3.

CG imagery has been successfully used to augment or even replace real training data for applications such as autonomous driving [46, 42], semantic segmentation [16, 44], and facial landmark detection [51]. There are now companies dedicated to generating CG images for training computer vision AI models [2, 4, 7]. Graphics platforms, such as Unreal [6], Unity [5], and Blender [1], typically render images directly to a device-independent display-referred color space—namely, standard RGB (sRGB). Moreover, computer graphics are implicitly white-balanced during rendering. Cameras also process RAW sensor images to sRGB. However, unlike graphics, a RAW image represents the sensor's direct response to physical radiance from the scene. RAW images are in a sensor-specific color space defined by the spectral sensitivities of the sensor's color filters. In addition, RAW images exhibit a strong color cast

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

due to scene illumination. Cameras have dedicated image signal processors (ISPs) that apply a series of operations to convert the RAW sensor image to the final sRGB image [20]. Among the ISP operations is a white-balance step to remove the color cast due to the scene illumination and a color space transformation step to convert the RAW image from its sensor-specific color space to the device-independent sRGB color space. For low-level vision tasks, such as illuminant estimation, that train DNN-models directly on RAW images, graphics images cannot be directly used. As a result, a mapping procedure explicitly targeting the conversion of graphics to RAW is needed.

Closely related to the *graphics to RAW* problem is the topic of RAW reconstruction from camera-rendered sRGB images. These methods aim to recover the RAW sensor image from the camera's output sRGB image. Data-driven methods, such as Cycle ISP [55] and Invertible ISP [52], suffer from the same data acquisition bottleneck we seek to avoid in that they require a large dataset of paired RAW-sRGB images from the target sensor for training. Metadata-assisted RAW reconstruction methods [53, 40, 39] assume that samples from the RAW image are available to reconstruct the RAW image and, therefore, cannot be applied to our task. The method of [10] is the closest to our work. This approach inverts ISP operations step-by-step to convert camera-rendered sRGB images back to plausible RAW images to be used for training image denoisers. However, [10] does not strive for color accuracy which is critical for other tasks such as illuminant estimation or color rendering.

Graphics images have several advantages over camera-rendered images. Camera sRGB images may have limited bit depth and resolution, and contain demosaicing or ringing artifacts, residual noise, and blur due to the ISP's RAW image processing. Properly rendered graphics images do not suffer from these issues and can produce significantly higher-quality training data. Furthermore, data diversity and concerns over privacy are less of an issue with CG.

Contributions We provide a description outlining how graphics images are directly rendered to sRGB and how cameras process RAW images to sRGB. Based on this insight, we present a framework that converts CG images to appear as RAW images captured by a target sensor under varied illuminations. In contrast to existing DNN methods that require large datasets of RAW-sRGB pairs for training, our approach requires only a handful of RAW-DNG files from the target sensor. We use the metadata in the DNG files to sample the illumination space of the target sensor and map the graphics images to a RAW color space that accurately mimics the target sensor. We can save our synthetic RAW images with the appropriate metadata back to DNG files so that they can be rendered by any standard DNG reader, such as Photoshop (see Fig. 1). We compare our results to competing methods and demonstrate the usefulness of our approach to DNN-based RAW image denoising, illumination estimation, and neural rendering. Finally, we provide a dataset of 292 nighttime graphics images for training neural ISPs targeting night photography based on synthetic RAW images. To our knowledge, this is the first work to target RAW image synthesis from CG imagery.

2. Related work

Related work is discussed regarding CG for high-level computer vision tasks, RAW recovery methods targeting camera images, and multi-spectral rendering.

Computer graphics in high-level vision tasks Graphics images have been used effectively as training data for a myriad of high-level computer vision tasks that are applied to camera-based sRGB images, such as autonomous driving [46, 42], semantic segmentation [16, 44], facial landmark detection [51], optical flow [21, 12], text localization [26], and object detection [27]. This has spawned large synthetic datasets focusing on specific high-level tasks (e.g, SHIFT [46], SYNTHIA [44], Sintel [12], Flying Chairs [21]). It is noteworthy that all of these datasets are comprised of sRGB images which are often not suitable as training data for low-level tasks that are applied to RAW sensor images. Allowing such images to be useful for tasks targeting RAW images is the motivation for our work.

RAW reconstruction from sRGB camera images Early work in the area of RAW reconstruction focused on radiometric calibration [19, 37, 25, 15, 14, 31]. However, these methods typically require tedious camera calibration procedures that have to be applied per camera. More recently, DNN-based RAW reconstruction methods have been proposed [55, 52, 34, 38] to model the imaging pipeline in both forward and reverse directions. Although these methods do not require calibration, a large number of RAW-sRGB image pairs from the target sensor are needed.

Another related research direction is methods that use specialized metadata to enable more accurate RAW reconstruction [53, 40, 39]. This metadata is gathered at capture time and is stored along with the sRGB image to facilitate RAW reconstruction post-capture. However, neither the DNN methods [55, 52, 34, 38] nor the metadata-assisted methods [53, 40, 39] can be applied to graphics data since there is no notion of an initial RAW sensor image.

A more practical approach, with minimal data requirements, is to reverse the sRGB image back to RAW by inverting the operations of the ISP stage-by-stage [10, 32]. It is assumed that a small set of RAW images is available to characterize the sensors' spectral response. For example, the unprocessing images (UPI) method [10] inverts the ISP stages by randomly sampling from distributions constructed using metadata extracted from the RAW files. However, UPI unprocesses images not to a specific sensor's RAW color space but to a plausible "meta" RAW color space produced by ag-

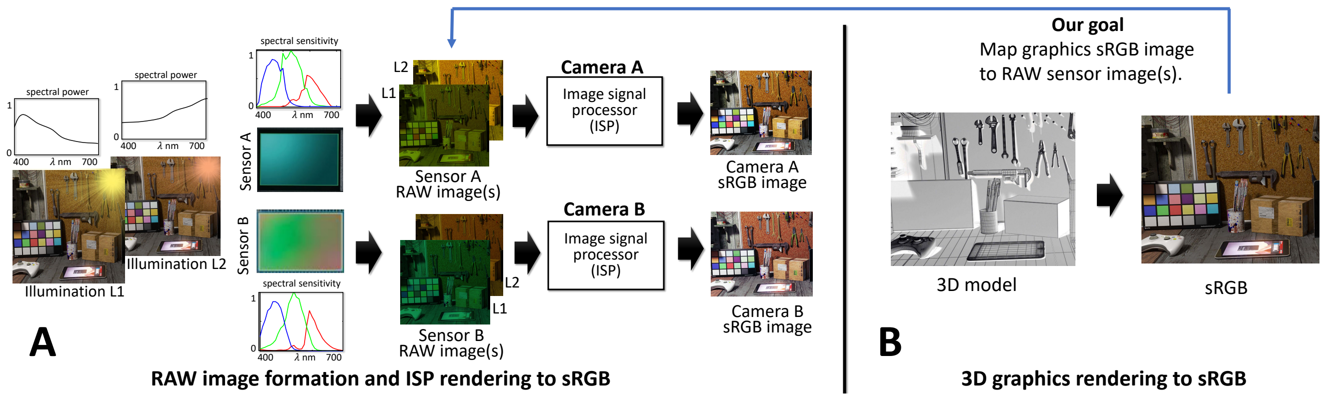

Figure 2. An illustration of sRGB image formation for cameras versus CG renderings. A camera's RAW sensor image formation is impacted by the scene's illumination and the spectral sensitivities of the color filters. Camera hardware ISPs process the RAW image to compensate for scene illumination and convert the image to the device-independent sRGB color space. In principle, images under different illumination should be rendered to visually similar sRGB images. Graphics images, on the other hand, are directly rendered to sRGB from the 3D models. Furthermore, graphics renderings do not require white-balancing. Our work aims to convert sRGB graphics images to a family of RAW images mimicking different illuminations and sensors.

gregating parameters from a mix of sensors. Hence, the RAW images produced by this procedure are not suitable for tasks such as illumination estimation and neural rendering, where accurate colors for a specific sensor are needed.

Our method converts a graphics sRGB image to a RAW image targeting a specific sensor by applying the inverse of the operations performed by an ISP. However, the difference in image formation between a graphics engine and a camera ISP means that only certain operations need to be applied to graphics sRGB images compared with camera sRGB images. We motivate and describe what these steps are. Moreover, unlike the sampling strategy of UPI [10], our sampling of the illumination and color correction matrices aligns closely with the operations on a real camera ISP, and leads to more accurate RAW color recovery.

Multi-spectral rendering Another approach to generate synthetic sensor RAW images is to directly render multi-spectral data using a supported CG engine. However, multi-spectral rendering is a significantly more complicated solution. Spectral information associated with all surface materials, illumination sources and texture maps need to be known. More importantly, the target sensor's spectral sensitivity also needs to be known; information that is not easily accessible (see Ch. 2.1.3.2 & 2.2.1 of [50]). Our method requires no specialized rendering or sensor calibration. Instead, we only require RAW DNGs from the target camera.

3. Graphics to RAW

Sec. 3.1 provides a high-level overview of camera versus graphics image formation. This is followed by a detailed description of how white-balancing and color mapping from RAW to sRGB are performed in Sec. 3.2. Understanding this process makes it clear how a graphics image can be reversed to a target sensor's RAW color space under a target illumination. Finally, in Sec. 3.3, we describe other camera rendering operations and their associated metadata. This is required to understand how a RAW-DNG file based on our synthesized RAW can be created.

3.1. Camera versus graphics image formation

We begin by reviewing the image formation on a camera and then compare it to that on a graphics engine. Each pixel on the camera sensor has a single R, G, B color filter arranged in a Bayer pattern. The spectral sensitivity of the color filter array (CFA) determines the sensor's response to incoming light. In particular, the pixel intensity at location x on the sensor image I_x is the product of the CFA's spectral sensitivity $\mathcal{F}$, the surface reflectance $\mathcal{P}$ of the material being imaged, and the spectral distribution of the illumination $\mathcal{L}$ in the scene, integrated over the visible spectrum Λ as follows:

$$I_x^c = \int_{\lambda \in \Lambda} \mathcal{L}_{x,\lambda} \mathcal{P}_{x,\lambda} \mathcal{F}_{c,\lambda}, \quad c \in \{R, G, B\}. \qquad (1)$$

If the illumination in the scene changes, the RAW pixel intensity value recorded by the sensor changes. Likewise, if a sensor with a different CFA is used, the RAW pixel intensity value changes. This is illustrated in Fig. 2. With modern smartphones being equipped with more than one camera, the second sensor may even be on the same device.

The camera's ISP is responsible for processing the RAW sensor image to the sRGB color space. The ISP processes the RAW image to remove the color cast from the scene illumination via a white-balancing operation, and transforms the image from a sensor-specific color space to the device-independent sRGB color space. Ideally, the ISP should produce the same canonical sRGB representation of the scene, irrespective of changes in illumination or the sensor. In practice, however, cameras apply additional proprietary photo-finishing routines to make the image more aesthetically appealing. As a result, the exact same scene may look visually different for different cameras [20].

The image formation for graphics images is significantly simpler. Graphics engines render directly to sRGB and have no notion of sensor spectral sensitivities or Bayer patterns. Moreover, CG images do not require white balancing—renderings are implicitly white-balanced. Graphics renderings typically do not include additional photo-finishing operations, such as tone manipulation, that need to be undone (see supplemental material for more details). Graphics do include a perceptual gamma encoding that is part of the sRGB standard. Some engines, such as Unreal, allow rendering to an ungamma sRGB, referred to as linear-sRGB. From Fig. 2, we see that a single graphics image can therefore be mapped to different illuminations under the same sensor or different sensors under the same illumination. To understand how to perform this mapping, we need to provide an overview of white-balance and color correction.

3.2. Camera white-balance and color correction

Unprocessed RAW camera images contain noticeable color cast due to the scene illumination. Cameras compensate for scene illumination using an auto-white-balance (AWB) procedure. AWB is a two step process that involves (i) estimating the color of the illumination in the camera sensor's RAW-RGB color space, and (ii) correcting the image based on the estimated illumination. Cameras estimate the illuminant from the image using an onboard AWB algorithm. While traditional AWB algorithms were based on image statistics [11, 22, 47, 30], DNN-based approaches have grown in popularity in recent years [28, 8, 9, 35]. The estimated illuminant is saved in the 'AsShotNeutral' (ASN) tag in the RAW DNG file. The second step of white-balance correction is as simple as multiplying the image by a 3×3 diagonal matrix constructed directly from the illumination parameters. This operation divides out the image per color channel by the scene illumination producing an image that appears to be lit under a neutral illumination.

After white balance has been performed, the color cast from the scene illumination has been removed. However, the white-balanced RAW image is still in a sensor-specific color space. Cameras apply a color transform to map the RAW image into a canonical perceptual color space, such as CIE XYZ. This is done using a full 3×3 matrix called the color space transform (CST) matrix. However, the CST is not a fixed matrix, but varies with the illumination. This is because white balance balances only the white/achromatic colors; non-neutral scene materials are not guaranteed to be properly corrected [18, 13]. Errors in non-neutral colors are dependent on the scene illumination. Therefore, to achieve better color reproduction, the CST matrix is computed by interpolating, based on the estimated illuminant, between two factory-calibrated CSTs computed for two fixed illuminations. These two fixed illuminants corresponding to the pre-calibrated matrices are selected to be far apart in terms of their correlated color temperature (CCT) values. D65 (daylight) and Standard light A (indoor-incandescent) are typically chosen as the illuminants on most cameras. These factory-calibrated matrices are part of the camera's firmware, and are recorded in the 'ColorMatrix1' and 'ColorMatrix2' tags in the DNG file. Their corresponding illuminants are stored in the 'CalibrationIlluminant1' and 'CalibrationIlluminant2' tags in the DNG. DNG-rendering software, such as Adobe Photoshop or Gimp, uses these tags in the DNG metadata to interpolate a CST matrix based on the estimated illuminant value recorded in the ASN tag.

Cameras often perform photofinishing operations in CIE XYZ or ProPhoto color spaces because they have a wider gamut than sRGB. Finally, they convert the photofinished image to the sRGB space. This step uses a fixed 3×3 'XYZ2sRGB' matrix. The combination of the CST and XYZ2sRGB matrix is often called the color correction matrix (CCM). As previously mentioned, CG images are rendered directly to sRGB without these intermediate steps and with an implicit white balance.

To convert CG images to RAW images from a specific sensor, we need to convert from sRGB space to the sensor's native color space, and apply a color cast assuming an illuminant in the scene. We achieve this with the help of a small set of RAW DNGs from the target sensor. We first extract the ASN tags and build a distribution around them to randomly sample illuminations from. Alternately, if the ground truth illumination in the scene is available, (e.g., as in the case of color constancy datasets, such as the NUS [17]), it is more accurate to use these instead of the ASN values. Based on the randomly sampled illuminant vector, we estimate the CST matrix by interpolating between the 'ColorMatrix1' and 'ColorMatrix2' tags in the DNG. Our method utilizes only the metadata in the DNG files (e.g., as-shot-neutral tags for illumination sampling, CST matrices) and does not look at image content. As such, we require images under different lighting conditions, regardless of the actual scene content.

Formally, let us assume we have M DNGs from the target sensor. We denote the chromaticity values ($\frac{R}{G}$ and $\frac{B}{G}$) of M ASN parameters extracted from these DNGs by the set $\mathcal{D}$. We fit a 2D multivariate Gaussian distribution of joint chromaticity values around the set $\mathcal{D}$ [41]. Then, we randomly sample an illuminant L from this distribution $\mathcal{N}$ as:

$$L \sim \mathcal{N}\left(\mu, \mathbf{\Sigma}\right) \,, \tag{2}$$

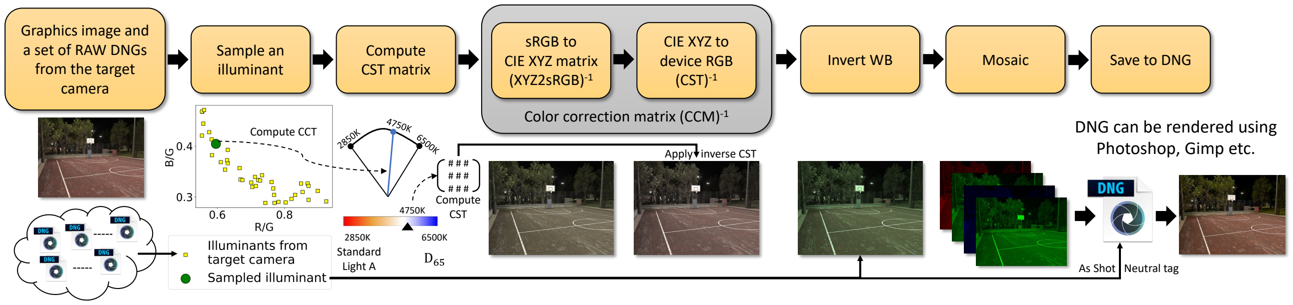
Graphics image and a set of RAW DNGs from the target camera
Sample an illuminant
Compute CST matrix
sRGB to CIE XYZ matrix (XYZ2sRGB)$^{-1}$
CIE XYZ to device RGB (CST)$^{-1}$
Color correction matrix (CCM)$^{-1}$
Invert WB
Mosaic
Save to DNG
DNG can be rendered using Photoshop, Gimp etc.
Compute CCT
B/G
0.4
0.3
0.6
0.8
R/G
2850K
4750K
6500K
Compute CST
Apply inverse CST
Illuminants from target camera
Sampled illuminant
Standard Light A
D$_{65}$
DNG
As Shot
Neutral tag

Figure 3. A visualization of our graphics to RAW processing pipeline. Given sRGB graphics images, and a set of RAW DNGs from the target sensor, we can convert the graphics images to realistic RAW images that mimic the illumination and spectral profile of the target sensor. We can also save our processed RAW image and associated metadata into a DNG file that can be rendered with standard DNG-rendering software.

$$\mathbf{\Sigma} = \frac{1}{M}\sum_{i=1}^{M}\left(\left(\left[\frac{R}{G},\frac{B}{G}\right]_i-\mu\right)^{\mathsf{T}}\left(\left[\frac{R}{G},\frac{B}{G}\right]_i-\mu\right)\right), \tag{3}$$

where μ and $\mathbf{\Sigma}$ are the mean and covariance of the normalized chromaticity values in $\mathcal{D}$, respectively, and $L, \mu \in \mathbb{R}^2$ and $\mathbf{\Sigma} \in \mathbb{R}^{2\times 2}$. Next, we compute the correlated color temperature corresponding to the sampled illumination L [13, 3]. Based on the estimated CCT value, the two pre-calibrated CSTs, S^A and S^{D65}, are interpolated as:

$$S^L = \alpha S^A + (1-\alpha)S^{\text{D65}}, \alpha \geq 0, \tag{4}$$

where

$$\alpha = \frac{\text{CCT}_L^{-1} - \text{CCT}_{\text{D65}}^{-1}}{\text{CCT}_A^{-1} - \text{CCT}_{\text{D65}}^{-1}}. \tag{5}$$

To process the graphics image, we first apply the inverse of the fixed XYZ2sRGB matrix followed by the inverse of the CST matrix (jointly called the CCM), and finally invert the white balance. While applying the inverse WB, we handle saturated pixel intensities using the highlight-preserving transform in [10]. Fig. 3 shows an overview of our method.

3.3. Other ISP operations and saving to DNG

In addition to WB and color correction, ISPs apply other operations to enhance the visual appeal of the image. While most operations need to be inverted when reverting from a camera-rendered sRGB to RAW, operations such as tone mapping typically are not applied to graphics sRGB images. Cameras apply a digital gain to the image based on the camera's auto exposure algorithm. We mimic this using a global scale factor sampled from a normal distribution, and applied along with the white balance inversion step [10]. Camera sensors have a Bayer layout, with each photosite housed beneath a single red, green, or blue color filter. Demosaicing is the process of estimating all three color values at each pixel location. To invert this step, we simply drop two of the three color values at each pixel location based on the Bayer pattern. To mimic sensor noise, we can add synthetic noise to the RAW image. The heteroscedastic Gaussian model [23, 33, 36] is the most widely used noise model. Heteroscedastic noise is modeled as a combination of shot and read parameters as $n_i \sim \mathcal{N}(0, \beta_1 x_i + \beta_2)$, where x_i, n_i are the clean, noisy pixel intensities at location i, and β_1 and β_2 are the shot and read noise parameters. However, more complex models (e.g., [49]) can also be used.

Our RAW image and associated metadata can be saved in the same DNG format as used by the target camera. Inside a DNG, RAW data is typically stored in Bayer format before black- and white-level adjustment, with the range of pixel intensity values determined by the sensor's bit depth. To save to DNG, we scale pixel intensities according to the target bit depth. Specifically, we denormalize the image based on the black- and white-level values as $\mathbf{I}_{dn} = \mathbf{I}(wl - bl) + bl$, where $\mathbf{I}_{dn}$ is the image $\mathbf{I}$ after denormalization, and bl, wl are the black- and white-level values, respectively. Using a DNG from the target sensor as a reference DNG, we write the denormalized RAW data to file. We also write our sampled illuminant into the ASN tag. Our graphics DNG can be rendered using any commercially available DNG-rendering software. In Fig. 1, (B) is our graphics DNG mimicking a Samsung S20 FE smartphone camera, while (A) is a real camera DNG from the Nikon D40 DSLR camera downloaded from the NUS dataset [17].

4. Experiments

We first show visual comparisons between our method and competing approaches in their ability to convert a graphics image to a RAW image, and then render the synthetic RAW image back to sRGB. We also evaluate our method's effectiveness on three tasks: (i) RAW image denoising, (ii) illumination estimation, and (iii) a neural ISP.

4.1. Datasets and our nighttime CG images

For the denoising and neural ISP experiments, we use the nighttime images from the dataset of [41]. This dataset con-

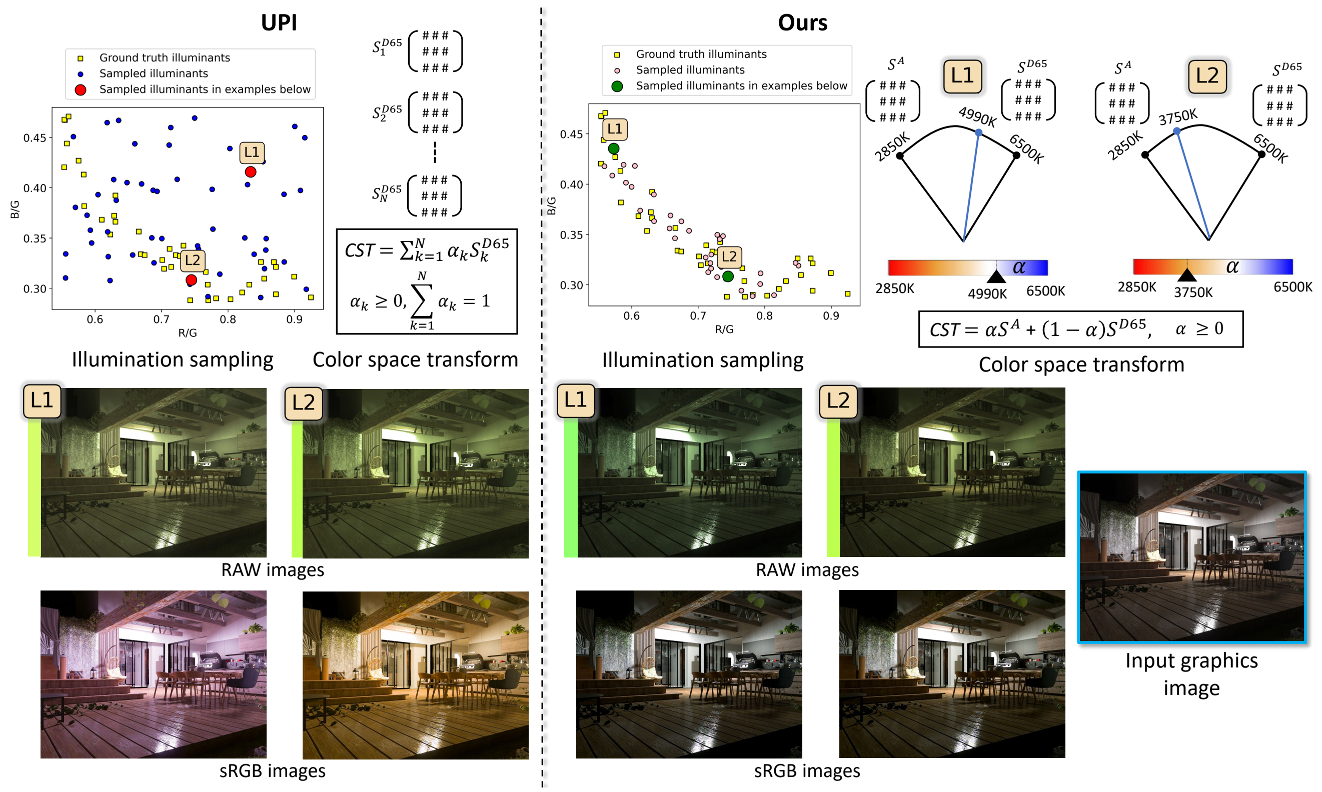

Figure 4. Comparison with UPI [10] on a CG image from our nighttime dataset. UPI samples the $\frac{R}{G}$, $\frac{B}{G}$ chromaticity values uniformly and independently, often resulting in samples that are very different from real illuminations. UPI's CST matrix is also sampled independently of the illumination. If there are N cameras $\{S_k\}_{k=1}^{N}$ in the target dataset, the CST matrix is computed as a random convex combination of the pre-calibrated CST matrices S_k^{D65}, $k = 1$ to N, corresponding to the D65 illuminant from the N cameras. We adopt a multivariate Gaussian sampling approach that closely follows the distribution of real illuminations. As performed on a real camera ISP, our CST matrix is computed based on the CCT value of the sampled illumination, by interpolating between the two pre-calibrated CST matrices, S^A and S^{D65}, corresponding to a specific target sensor S. The vertical bar to the left of the images represents the color of the illuminant in the RAW space – L1 and L2 represent illuminant samplings by UPI and our method. For this example, the L2 sample is the same for both methods. When the synthetic RAW images are rendered to sRGB using Photoshop, our method produces more natural colors.

tains 105 real low-light nighttime scenes captured in RAW format using a Samsung S20 FE smartphone camera. To our knowledge, a high-quality nighttime CG dataset is not available. To address this, we generated a dataset containing 292 nighttime CG images. We used the Unreal Engine 5 [6] with path tracing to produce highly realistic images, and output data in 32-bit floating point EXR format (see supplemental materials for examples).

For the illumination estimation task, we use the well-established NUS color constancy dataset [17]. The NUS dataset has RAW images captured using nine different DSLR cameras. For illumination estimation, we apply our method on CG images from the SYNTHIA [44] dataset.

4.2. Comparison methods and visual results

Our closest competitor is the RAW unprocessing method of UPI [10]. UPI assumes that the metadata from the target sensors is available. In particular, the white-balance and digital gain ranges, color correction matrices, and noise parameters for the target sensors have to be specified to their algorithm. The parameters are assimilated over the pool of target sensors. We use the official implementation from the authors, and replace the parameters appropriately by selecting the cameras from the NUS dataset [17] and the nighttime dataset of [41] as the target sensors (since those are the datasets we have selected for our experiments). UPI samples the $\frac{R}{G}$, $\frac{B}{G}$ white-balance gain values uniformly and independently to apply the illumination color cast. The color space transform is computed as a random convex combination of the pre-calibrated color space transforms corresponding to the D65 illuminant from all the target sensors. This convex combination is computed independent of the sampled illuminant. This is shown in the left column of Fig. 4. Notice that our sampling approach, shown on the right, is more realistic, and closely follows the processing steps on an actual ISP—our multivariate Gaussian sampling of the illumination produces samples close to the ground truth illuminations, and the CST matrix is computed, based

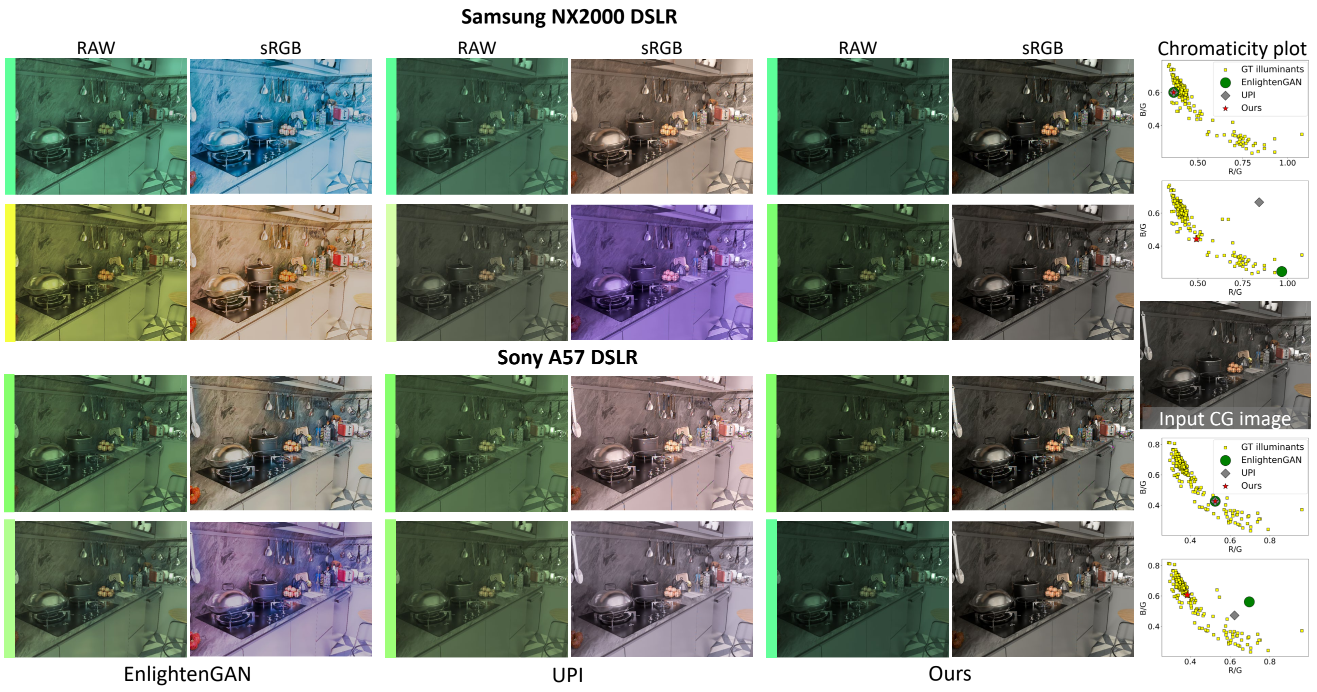

Figure 5. A visual comparison between EnligtenGAN [29], UPI [10], and our method. Two DSLR cameras from the NUS dataset [17] are selected as the target sensors. The input image is from our CG dataset. The vertical bar to the left of the images represents the color of the illuminant in the RAW space, with two random illuminants shown per sensor for each method. For both sensors, the illuminant in the first example is chosen to be the same for EnlightenGAN [29], UPI [10], 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.

on the sampled illuminant, by interpolating between the two pre-calibrated CST matrices for that specific sensor. The RAW images corresponding to two different randomly sampled illuminants L1, L2 are shown in the figure, with L2 being the same for both UPI and our method. The vertical bar on the left of each image shows the color of the illuminant in the RAW space. For both methods, we save the RAW images to DNG format, and replace the ASN tag with the sampled illumination. We use the Samsung S20 FE main rear camera's DNG from the dataset of [41] for this example. The bottom row of the figure shows Adobe Photoshop's sRGB rendering of the DNG files with an auto white balance applied. It can be clearly observed that our method produces a more natural color rendering that resembles the original graphics image. Although the two sampled illuminants have very different RAW color casts, applying Photoshop's AWB correction on our RAW DNGs produces nearly identical results. This is in line with our expectation that the rendered output should be independent of the environmental illumination. However, with Photoshop's AWB applied, UPI's strategy produces undesirable color casts in the rendered sRGB images, even for the illuminant L2 which lies close to the distribution of real illuminants.

We also compare against an unpaired generative approach using the well-known EnlightenGAN model [29]. We train an EnlightenGAN model per camera to map graphics sRGB images to RAW images from the target sensor. Since a single graphics image can map to many RAW images with different illumination color casts, we train EnlightenGAN to map sRGB images to white-balanced RAW images. The color cast is applied as a post-process step to the output of EnlightenGAN by inverting the WB, with the illuminant sampled from the illumination space of the target sensor following the approach of UPI. We show results of EnlightenGAN in the first column of Fig. 5 with two DSLR cameras from the NUS dataset [17] selected as the target sensors. Two different illuminations (with their colors represented by the vertical bars) are shown for each sensor. As before, the RAW images are saved to DNG format with the ASN tag set to the sampled illuminant. The sRGB images are obtained using Photoshop with the AWB setting applied. It can be observed that there are unnatural color artifacts in the rendered output of EnlightenGAN. Fig. 5 also shows the results of UPI and our method in the second and third columns, respectively. Once again, our method produces more natural colors. See supplemental for more results.

4.3. RAW denoising

We use the nighttime dataset of [41] for our RAW image denoising experiment. Noisy RAW images at ISOs 1600

Table 1. RAW image denoising results on the nighttime dataset of [41].

Model	ISO 1600		ISO 3200	
	PSNR	SSIM	PSNR	SSIM
EnlightenGAN [29]	48.82	0.9906	47.25	0.9872
UPI [10]	49.05	0.9904	47.51	0.9873
Ours	**49.37**	**0.9911**	**48.16**	**0.9892**
Real	49.80	0.9927	48.25	0.9899

and 3200, and their corresponding clean RAW images are available as part of the dataset. We choose the Restormer model in the recent work of [54] as our denoiser. To generate synthetic RAW images, we use our nighttime graphics dataset. We generate "clean" ground-truth RAW data by applying our method to these images. The dataset of [41] also provides a calibrated noise model for the S20 FE main rear camera, with which the dataset was captured. The calibration procedure assumes the standard heteroscedastic Gaussian noise model. We use their noise generator to generate synthetic noisy RAW images at ISOs 1600 and 3200. Similarly, we generate noisy/clean RAW image pairs by applying UPI and EnligtenGAN on the same graphics data and using the same noise generator. For comparison, we also train a supervised model exclusively on real data from [41]. Additional implementation details are provided in the supplementary material. PSNR (dB) and SSIM [48] values averaged over the 105 real images at ISOs 1600 and 3200 in the dataset of [41] are reported in Table 1. We outperform both EnlightenGAN and UPI while being on par with the model trained on real data.

4.4. Illuminant estimation

Next, we examine the task of illuminant estimation using the NUS dataset [17]. Since the NUS images are mostly outdoor or well-lit indoor scenes, we do not use our nighttime graphics data. Instead, we use the SYNTHIA dataset [44]. We generate synthetic RAW data from the CG images in this dataset using EnlightenGAN, UPI, and our method. We use a lightweight CNN architecture from [24] as our illuminant estimation network. Network architecture as well as training and testing details can be found in the supplementary material. Angular errors averaged over all nine cameras from the NUS dataset [17] are presented in Table 2. Our approach outperforms both EnlightenGAN and UPI by a significant margin, and has the least gap with the models trained on real data.

4.5. Neural ISP

For our final task of a neural ISP, we once again use the nighttime images from the dataset of [41]. To study the color reproduction accuracy in isolation, independent of the effects of noise, we examine the first scenario described in [41] of a neural ISP that renders a clean nighttime RAW

Table 2. Illuminant estimation results on the NUS dataset [17]. Angular errors are reported.

Model	Mean	Median	Top 25%	Worst 25%
EnlightenGAN [29]	7.01	6.82	3.48	11.07
UPI [10]	6.26	5.89	2.92	10.33
Ours	**4.21**	**3.38**	**1.30**	**8.57**
Real	3.02	2.17	0.75	6.77

Table 3. Quantitative results on our neural ISP task on the nighttime dataset of [41]. A lower ΔE [45] is better.

Model	PSNR	SSIM	ΔE [45]
EnlightenGAN [29]	35.58	0.965	3.137
UPI [10]	36.43	0.966	2.907
Ours	**38.10**	**0.974**	**2.301**
Real	38.32	0.974	2.133

image to sRGB. In their work, the sRGB images are generated using a simplified Python-based software ISP. For uniformity in the software ISP used throughout our experiments, we use Photoshop to render their RAW images to sRGB. A UNet [43] architecture is used as the ISP model in [41]. We use the official implementation from the authors for training and testing. We use our nighttime dataset to generate synthetic training data for our method as well as EnligtenGAN and UPI. More details and qualitative comparisons are provided in the supplementary material. Quantitative results are presented in Table 3. Our method performs on par with the model trained on real data, while our closest competitor UPI produces a PSNR value that is a significant 1.6 dB lower than our result.

5. Conclusion

We have presented a framework for converting computer graphics images to mimic RAW sensor images. As part of this work, we described how graphics images are directly rendered to sRGB and how cameras process RAW images to sRGB. This understanding allowed us to propose a method to convert a graphics image to any target sensor's RAW color space, while emulating the RAW image under different illuminations. Compared to methods targeting camera sRGB images, our approach produces more visually realistic results. Moreover, our proxy experiments on three low-level vision tasks—RAW image denoising, illumination estimation, and night photography rendering—show our method provides more accurate synthetic RAW data. To test the night photography task, we generate 292 realistic CG images of night and low-light scenes that we will make publicly available along with our code for graphics2RAW conversion. We believe this work will be another useful tool in reducing the burden of data collection via computer graphics imagery.

References

[1] Blender. https://www.blender.org. Accessed: 2022-11-06. 1

[2] Datagen. https://datagen.tech. Accessed: 2022-11-06. 1

[3] DNG Specification. https://helpx.adobe.com/content/dam/help/en/photoshop/pdf/dng_spec_1_6_0_0.pdf. Accessed: 2022-11-09. 5

[4] Synthesis AI. https://synthesis.ai/. Accessed: 2022-11-06. 1

[5] Unity. https://unity.com/. Accessed: 2022-11-06. 1

[6] Unreal Engine. https://www.unrealengine.com/unreal-engine-5. Accessed: 2022-11-06. 1, 6

[7] Zumo Labs. https://www.zumolabs.ai/. Accessed: 2022-11-06. 1

[8] Jonathan T Barron and Yun-Ta Tsai. Fast Fourier color constancy. In *CVPR*, 2017. 4

[9] Simone Bianco and Claudio Cusano. Quasi-unsupervised color constancy. In *CVPR*, 2019. 4

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

[11] Gershon Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin institute*, 310(1):1–26, 1980. 4

[12] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012. 2

[13] Hakki Can Karaimer and Michael S Brown. Improving color reproduction accuracy on cameras. In *CVPR*, 2018. 4, 5

[14] Ayan Chakrabarti, Daniel Scharstein, and Todd E. Zickler. An empirical camera model for internet color vision. In *BMVC*, 2009. 2

[15] Ayan Chakrabarti, Ying Xiong, Baochen Sun, Trevor Darrell, Daniel Scharstein, Todd Zickler, and Kate Saenko. Modeling radiometric uncertainty for vision with tone-mapped color images. *TPAMI*, 36(11):2185–2198, 2014. 2

[16] Yuhua Chen, Wen Li, Xiaoran Chen, and Luc Van Gool. Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach. In *CVPR*, 2019. 1, 2

[17] 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. 4, 5, 6, 7, 8

[18] Dongliang Cheng, Brian Price, Scott Cohen, and Michael S. Brown. Beyond white: Ground truth colors for color constancy correction. In *ICCV*, 2015. 4

[19] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH*, 2008. 2

[20] Mauricio Delbracio, Damien Kelly, Michael S. Brown, and Peyman Milanfar. Mobile computational photography: A tour. *Annual Review of Vision Science*, 7(1):571–604, 2021. 2, 4

[21] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 2

[22] Graham D Finlayson and Elisabetta Trezzi. Shades of gray and colour constancy. In *Color and Imaging Conference*, 2004. 4

[23] Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data. *TIP*, 17(10):1737–1754, 2015. 5

[24] Han Gong. Convolutional mean: A simple convolutional neural network for illuminant estimation. In *BMVC*, 2019. 8

[25] Michael D. Grossberg and Shree K. Nayar. Determining the camera response from images: What is knowable? *TPAMI*, 25(11):1455–1467, 2003. 2

[26] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *CVPR*, 2016. 2

[27] Stefan Hinterstoisser, Olivier Pauly, Hauke Heibel, Marek Martina, and Martin Bokeloh. An annotation saved is an annotation earned: Using fully synthetic training for object detection. In *ICCV Workshop*, 2019. 2

[28] Yuanming Hu, Baoyuan Wang, and Stephen Lin. FC4: Fully convolutional color constancy with confidence-weighted pooling. In *CVPR*, 2017. 4

[29] 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. 7, 8

[30] Hamid Reza Vaezi Joze, Mark S Drew, Graham D Finlayson, and Perla Aurora Troncoso Rey. The role of bright pixels in illumination estimation. In *Color and Imaging Conference*, 2012. 4

[31] Seon Joo Kim, Hai Ting Lin, Zheng Lu, Sabine Süsstrunk, Stephen Lin, and Michael S. Brown. A new in-camera imaging model for color computer vision and its application. *IEEE TPAMI*, 34(12):2289–2302, 2012. 2

[32] Samu Koskinen, Dan Yang, and Joni-Kristian Kämäräinen. Reverse imaging pipeline for raw RGB image augmentation. In *ICIP*, 2019. 2

[33] Xinhao Liu, Masayuki Tanaka, and Masatoshi Okutomi. Practical signal-dependent noise parameter estimation from a single noisy image. *TIP*, 23(10):4361–4371, 2014. 5

[34] Yu-Lun Liu, Wei-Sheng Lai, Yu Sheng Chen, Yi-Lung Kao, Ming-Hsuan Yang, Yung-Yu Chuang, and Jia-Bin Huang. Single-image HDR reconstruction by learning to reverse the camera pipeline. In *CVPR*, 2020. 2

[35] Yi-Chen Lo, Chia-Che Chang, Hsuan-Chao Chiu, Yu-Hao Huang, Chia-Ping Chen, Yu-Lin Chang, and Kevin Jou. CLCC: Contrastive learning for color constancy. In *CVPR*, 2021. 4

[36] Markku Mäkitalo and Alessandro Foi. Optimal inversion of the generalized Anscombe transformation for Poisson-Gaussian noise. *TIP*, 22(1):91–103, 2013. 5

[37] Tomoo Mitsunaga and Shree K. Nayar. Radiometric self calibration. In *CVPR*, 1999. 2

[38] Seonghyeon Nam and Seon Joo Kim. Modelling the scene dependent imaging in cameras with a deep neural network. In *ICCV*, 2017. 2

[39] Seonghyeon Nam, Abhijith Punnappurath, Marcus A. Brubaker, and Michael S. Brown. Learning sRGB-to-raw-RGB de-rendering with content-aware metadata. In *CVPR*, 2022. 2

[40] Rang M. H. Nguyen and Michael S. Brown. RAW image reconstruction using a self-contained sRGB-JPEG image with only 64 KB overhead. In *CVPR*, 2016. 2

[41] 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. 4, 5, 6, 7, 8

[42] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 1, 2

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015. 8

[44] 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, 6, 8

[45] Gaurav Sharma and Raja Bala. *Digital Color Imaging Handbook*. CRC Press, 2nd edition, 2013. 8

[46] Tao Sun, Mattia Segu, Janis Postels, Yuxuan Wang, Luc Van Gool, Bernt Schiele, Federico Tombari, and Fisher Yu. SHIFT: A synthetic driving dataset for continuous multi-task domain adaptation. In *CVPR*, 2022. 1, 2

[47] Joost Van De Weijer, Theo Gevers, and Arjan Gijsenij. Edge-based color constancy. *TIP*, 16(9):2207–2214, 2007. 4

[48] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *TIP*, 13(4):600–612, 2004. 8

[49] Kaixuan Wei, Ying Fu, Jiaolong Yang, and Hua Huang. A physics-based noise formation model for extreme low-light raw denoising. In *CVPR*, 2020. 5

[50] Andrea Weidlich, Alex Forsythe, Scott Dyer, Thomas Mansencal, Johannes Hanika, Alexander Wilkie, Luke Emrose, and Anders Langlands. Spectral imaging in production. In *ACM SIGGRAPH Courses*, 2021. 3

[51] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Thomas J. Cashman, and Jamie Shotton. Fake it till you make it: Face analysis in the wild using synthetic data alone. In *ICCV*, 2021. 1, 2

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

[53] Lu Yuan and Jian Sun. High quality image reconstruction from RAW and JPEG image pair. In *ICCV*, 2011. 2

[54] 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. 8

[55] 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