Time-Aware Auto White Balance in Mobile Photography

Mahmoud Afifi[*] Luxi Zhao[*] Abhijith Punnappurath
Mohammed A. Abdelsalam Ran Zhang Michael S. Brown
AI Center–Toronto, Samsung Electronics
{m.afifi1, lucy.zhao, abhijith.p, m.abdelsalam, ran.zhang, michael.b1}@samsung.com

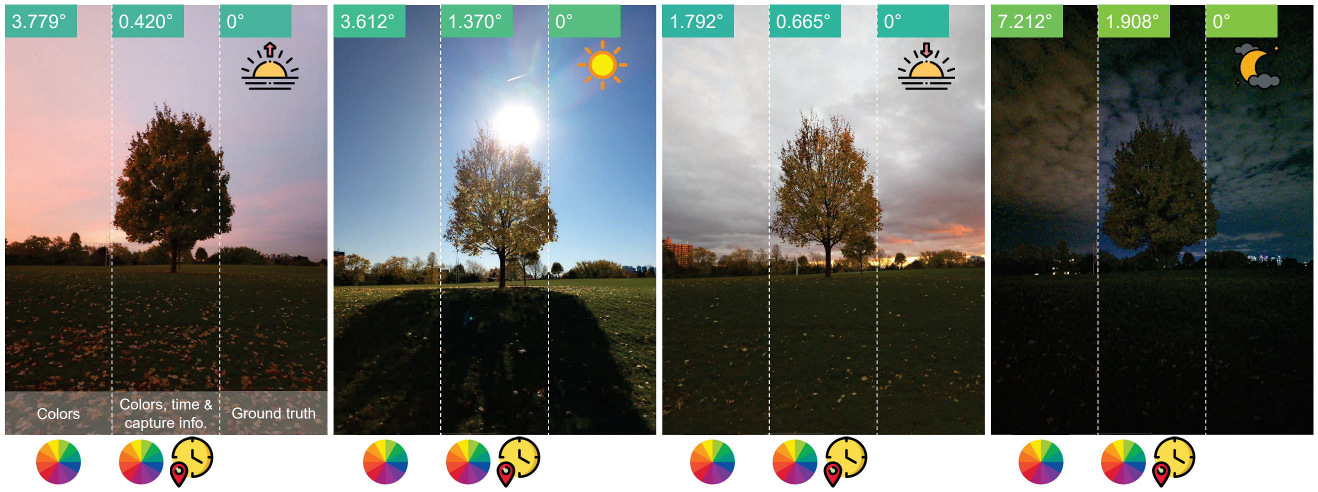

Figure 1. The time of day influences scene illumination, making it a valuable cue for improving illuminant estimation. Shown are white-balanced results (gamma-corrected for visualization) using our method with (1) colors only, (2) colors plus contextual (timestamp and geolocation) and capture data, and (3) ground truth (from a color chart). Angular errors show improvements when time-capture information is used.

Abstract

Cameras rely on auto white balance (AWB) to correct undesirable color casts caused by scene illumination and the camera's spectral sensitivity. This is typically achieved using an illuminant estimator that determines the global color cast solely from the color information in the camera's raw sensor image. Mobile devices provide valuable additional metadata—such as capture timestamp and geolocation—that offers strong contextual clues to help narrow down the possible illumination solutions. This paper proposes a lightweight illuminant estimation method that incorporates such contextual metadata, along with additional capture information and image colors, into a lightweight model ($\sim$5K parameters), achieving promising results, matching or surpassing larger models. To validate our method, we introduce a dataset of 3,224 smartphone images with contextual metadata collected at various times of day and under diverse lighting conditions. The dataset includes ground-truth illuminant colors, determined using a color chart, and user-preferred illuminants validated through a user study, providing a comprehensive benchmark for AWB evaluation.

[*]Equal contribution.

1. Introduction and related work

Color constancy refers to the ability of the human visual system to maintain stable object colors despite variations in lighting conditions by leveraging contextual cues within the scene [27, 33, 51]. Cameras approximate this effect using auto white balance (AWB) correction, which aims to partially neutralize color casts introduced by scene illumination and the camera's spectral sensitivity [3]. AWB first estimates the illumination color as an RGB vector in the camera's raw color space. The raw image is then corrected by scaling its color channels according to the estimated illumination, typically under the assumption of a single global light source [8, 31].

Conventional illuminant estimation methods primarily rely on image colors, either by directly processing the raw image (e.g., [13, 32, 41, 49, 53, 58, 62, 66]) or by analyzing color histograms (e.g., [2, 6, 7, 10]). These methods can be broadly categorized into two groups: (1) classical statistical-based methods (e.g., [15, 26, 32, 56, 57, 62]),

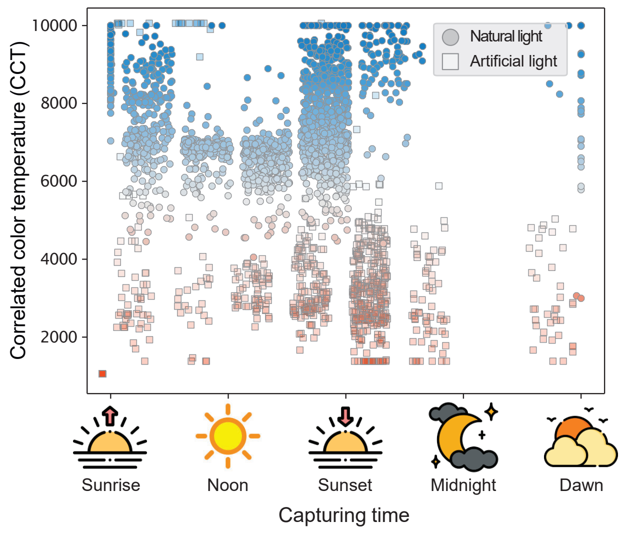

Figure 2. The time of day at which an image is captured provides valuable information about the possible range of illuminant colors in outdoor scenes. The figure presents the correlated color temperature (CCT) of illuminant colors in our dataset (Sec. 3) for images captured at various times throughout the day and night. As shown, excluding images taken under artificial light, those captured at noon, for example, exhibit a different range of illuminant CCTs compared to images captured during sunset or sunrise.

which estimate the illuminant color based on image statistics, and (2) learning-based methods (e.g., [6, 10, 30, 42, 59]), which map image colors to their corresponding scene illuminant through data-driven models.

While image colors are a key cue for estimating the scene's illuminant, mobile devices provide an opportunity to integrate additional contextual information. For instance, the device's location, along with the date and time, offer valuable cues about outdoor lighting conditions (e.g., sunrise, noon, sunset), thereby improving illuminant estimation for outdoor scenes (see Fig. 1).

Intuitively, knowing the time of day when an outdoor scene is captured can help estimate the lighting conditions. Figure 2 further illustrates that the time of day, derived from contextual metadata (i.e., timestamp and geolocation) readily available on mobile devices, provides valuable insights into the likely range of illuminant correlated color temperature (CCT) in outdoor scenes. This information serves to narrow the range of possible illuminant colors. For instance, images taken at noon exhibit a different CCT range than those captured at sunrise or sunset. When combined with additional capture information to distinguish between environments (e.g., indoors vs. outdoors), such metadata can complement image colors to improve illuminant estimation accuracy.

Despite its potential, only a few attempts have explored leveraging additional information available to the camera's image signal processor (ISP) for illuminant estimation, such as metadata-based model control [11] or data augmentation to enhance generalization [6]. However, to our knowledge, no previous work has investigated using contextual information—specifically, mobile device timestamps and geolocation—to refine illuminant estimation.

We introduce a method that leverages contextual metadata from mobile phones, along with additional capture information available in camera ISPs, to train a lightweight illuminant estimator model with $\sim$5K parameters. Our model delivers promising results, matching or surpassing that of larger models, while maintaining efficiency. Our model runs on a typical flagship mobile digital signal processor (DSP) and CPU in 0.25 ms and 0.80 ms, respectively. This compact and efficient design is especially advantageous for mobile devices, where minimizing power consumption and memory usage is critical [1, 7, 11].

Existing white-balance datasets (e.g., [19, 23, 28, 43]) lack the contextual information needed to validate our method. The absence of contextual information is because most available datasets (e.g., [19, 23]) used DSLR cameras, which typically lack built-in GPS functionality or accurate timestamps. To address this, we captured a new dataset of 3,224 images using a consumer smartphone camera, accompanied by contextual and capture information. The ground truth for the dataset was established using two approaches: (1) the conventional approach, selecting gray patches from a calibration color chart, and (2) a manually selected user-preference white-balance target, which enhances the image's aesthetic appeal and accounts for incomplete chromatic adaptation [60]. The user-preference white-balance "ground truth" was validated through a user study. The dataset spans a variety of lighting conditions, including non-standard artificial illuminants [61], and covers different times of day—sunrise, noon, sunset, and night.

Contribution: We propose an AWB method that utilizes smartphone contextual metadata (i.e., timestamp and geolocation) alongside capture information to enhance illuminant estimation. We demonstrate that integrating this additional data with conventional color information into a lightweight neural model leads to significant improvements in accuracy. Additionally, we introduce a large-scale dataset captured with a consumer smartphone at various times of day, with ground truth derived from both a calibration color chart and user-preference-based white balance. Benchmarking results show that our method achieves state-of-the-art results in most standard metrics (i.e., angular error statistics [19]), while maintaining computational efficiency on mobile devices.

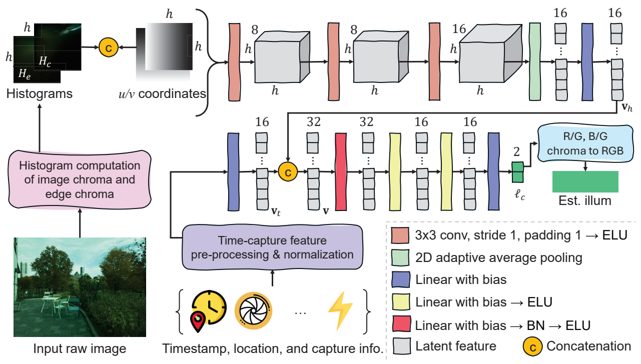
h
H_c
H_e
Histograms
u/v coordinates
C
8
16
32
2
$\mathbf{v}_h$
$\mathbf{v}_t$
$\mathbf{v}$
ℓ_c
R/G, B/G chroma to RGB
Est. illum
Histogram computation of image chroma and edge chroma
Time-capture feature pre-processing & normalization
Input raw image
Timestamp, location, and capture info.
3x3 conv, stride 1, padding 1 → ELU
2D adaptive average pooling
Linear with bias
Linear with bias → ELU
Linear with bias → BN → ELU
Latent feature
Concatenation

Figure 3. Our method includes a lightweight model consisting of a convolutional network that processes the histogram feature (derived from raw image colors and edge histograms concatenated with the u/v coordinates) to produce a latent feature $\mathbf{v}_h \in \mathbb{R}^{16}$. This is then concatenated with the latent feature of the processed time-capture feature, $\mathbf{v}_t \in \mathbb{R}^{16}$. The combined feature, $\mathbf{v} \in \mathbb{R}^{32}$, is passed through a lightweight MLP to produce the chromaticity of the scene illuminant, $\ell_c \in \mathbb{R}^2$, which is finally converted into normalized RGB illuminant color.

2. Method

An overview of our method is shown in Fig. 3. Our method employs a learnable model that processes two distinct inputs: (1) the time-capture feature (Sec. 2.1), which combines the contextual and capture information available on mobile devices and accessible by their camera ISPs, and (2) the histogram feature (Sec. 2.2), which represents the image's R/G and B/G chromaticity values. The time-capture feature is first processed to project it into a latent space, producing the time-capture latent feature vector, $\mathbf{v}_t \in \mathbb{R}^{16}$. The model processes the histogram feature to produce the histogram's latent feature vector, $\mathbf{v}_h \in \mathbb{R}^{16}$. Both the histogram and time-capture latent feature vectors are concatenated and processed by a set of learnable layers to output an R/G, B/G chromaticity vector, $\ell_c \in \mathbb{R}^2$, representing the scene illuminant. This 2D vector is then converted into the illuminant RGB color used to perform white balancing.

2.1. Time-capture feature

Our model incorporates contextual metadata, available on mobile devices, as one of its input features. Specifically, we use the geolocation and timestamp of image capture to compute the "probability" of the time of day (e.g., sunset, noon, etc.). Our approach converts conventional clock time (hour::minutes) to its corresponding *time of day* condition based on the date and geolocation of the scene. This approach allows the model to generalize across different time zones and prevents it from being influenced by the capturing location. The time probability vector represents the likelihood that the captured image corresponds to one of the solar event times (i.e., dawn, sunrise, noon, sunset, dusk, or midnight) and is computed as follows:

$$p_g = 1 - \frac{|t_c - t_g|}{t_s}, \tag{1}$$

where t_c is the capture time in seconds (adjusted to the local time zone based on geolocation information), and t_g is the local time in seconds of the solar event, $g \in \{\texttt{dawn}, \texttt{sunrise}, \texttt{noon}, \texttt{sunset}, \texttt{dusk}, \texttt{midnight}\}$, computed using geolocation-based standard algorithms [52, 63, 64]. The scalar t_s represents the total number of seconds in a day (i.e., 86,400).

We pre-process the probability of each solar event, p_g, in our time probability vector by computing the square root to enhance feature representation—compressing high probabilities and amplifying lower ones, which is intended to help create a more balanced and smooth distribution for the model to leverage. We then augment this time probability vector with a one-hot vector, $\mathbf{b}$, that indicates whether the capture time t_c occurs before each solar event. This distinction helps the model account for expected variations in CCT, as illuminant colors can differ before and after certain solar events, such as sunset and sunrise. The value of this one-hot vector for a given solar event g is computed as:

$$\mathbf{b}_g = \begin{cases} 1, & \text{if } t_c \le t_g \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where $\mathbf{b}_g$ corresponds to the entry in the one-hot vector for the solar event, g. Both the time probability vector and the one-hot vector together form our time feature, $\mathbf{p} \in \mathbb{R}^{12}$.

To enrich our feature set with additional capture information available from the camera ISP and help distinguish the capturing environment (e.g., indoor vs. outdoor), we include the following features in our final time-capture feature, $\mathbf{c}$:

- **ISO** (i): The sensitivity of the camera's image sensor to light, where lower values indicate good lighting conditions (e.g., bright scenes) and higher values suggest low-light environments, such as poorly lit indoor scenes.
- **Shutter speed** (s): The amount of time the camera's shutter remains open, allowing light to hit the image sensor. It provides an indication of lighting conditions, alongside the ISO value, i.
- **Flash status** (f): A binary value indicating whether flash light was used during capturing.
- **Noise information (optional)**: Since image denoising is typically applied before or in parallel with illuminant estimation in camera ISPs [22, 34, 45], we also explore the optional use of explicit noise information from the captured scene. More noise typically indicates low-light conditions, which can provide clues about the lighting color range of the scene. Unfortunately, while noise information is accessible within camera ISPs, obtaining accurate

noise information for public use is challenging, as the noise profiles in DNG files are not always reliable [68]. To address this, we simulate the noise information using two approaches: noise statistics (stats) and/or signal-to-noise ratio (SNR) stats, which are described below.

Noise stats (n): This represents the noise statistics in the captured raw image. We simulate this by denoising each raw image using Adobe Lightroom and computing the noise stats as the mean and standard deviation of each color channel of the absolute difference between the denoised and noisy raw images. This approach provides a simplified method for estimating noise stats, as the denoised images are typically available within the camera ISPs, but difficult to extract from the DNG files.

SNR stats (r): This alternative approach measures the noise information in the captured image without the need of a denoised reference. The SNR is computed by applying a 15×15 sliding window over the raw image and calculated as: $10 \log_{10} \left(\frac{\mu}{\sigma+\epsilon}\right)$, where μ represents the mean RGB of the 15×15 patch, σ is the standard deviation, and ϵ is a small value added for numerical stability.

Note that while these two approaches—namely, computing noise stats and SNR stats—yield satisfactory results, there is a wide body of research on noise estimation (e.g., [47, 54, 55]), which is beyond the scope of this paper.

Our complete time-capture feature is the combination of these inputs and can be expressed as follows:

$$\mathbf{c} = \left[\mathbf{p}^T, \log(i), \log(s), f, \mathbf{w}^T\right]^T, \tag{3}$$

where $\mathbf{w}$ represents optional noise-related features, which can include either noise stats $\mathbf{n}$, SNR stats $\mathbf{r}$, or both. Alternatively, $\mathbf{w}$ can be omitted if noise information is not considered.

Our time-capture feature is first normalized using min-max normalization (with the min and max values computed from the training data), and then processed through a learnable function, f_t, as follows:

$$\mathbf{v}_t = f_t(\mathbf{c}), \tag{4}$$

where f_t is a learnable linear layer that transforms the time-capture feature, $\mathbf{c}$, into its latent representation, $\mathbf{v}_t \in \mathbb{R}^{16}$ (see Fig. 3).

2.2. Histogram feature

In addition to the time-capture feature, $\mathbf{c}$, we provide our neural model with a histogram feature, $\mathbf{H}$, which represents the colors of the raw image. Inspired by prior work [6, 10, 11], we use a 2D histogram to represent the R/G and B/G chromaticities of the input raw image, $\mathbf{I} \in \mathbb{R}^{K \times 3}$, where K denotes the total number of pixels in the raw image. Specifically, we compute the 2D chromaticity histogram, $\mathbf{H}_c \in \mathbb{R}^{h \times h}$, as follows:

$$\mathbf{H}_c^{(m,n)} = \sum_k \|\mathbf{I}^{(k)}\|_2 \cdot \delta_{m,n}^{(k)}, \tag{5}$$

$$\delta_{m,n}^{(k)} = \left[u_m \leq rg^{(k)} < u_{m+1}\right] \wedge \left[v_n \leq bg^{(k)} < v_{n+1}\right], \tag{6}$$

where $rg^{(k)}$ and $bg^{(k)}$ are the R/G and B/G chromaticity values of pixel k in $\mathbf{I}$, and $\|\mathbf{I}^{(k)}\|_2$ represents the intensity of pixel k (i.e., Euclidean norm of the pixel's RGB values). The notation $\wedge$ denotes the logical AND operator, and $[\cdot]$ is the Iverson bracket, which evaluates to 1 when the condition is true and 0 otherwise. The histogram bins are defined by the edges $\{u_m\}$ and $\{v_n\}$, where u_m and v_n are the lower edges of the bins, and u_{m+1} and v_{n+1} are the corresponding upper edges. The histogram accumulates the brightness values of pixels whose chromaticity values fall within the range $[u_m, u_{m+1})$ along the R/G axis (i.e., the u-axis) and $[v_n, v_{n+1})$ along the B/G axis (i.e., the v-axis). The number of histogram bins along each axis is denoted as h. Following [10], we compute the square root of the histogram to enhance the utility of the histogram feature [9].

Note that this histogram differs from the *log-uv* histogram used in prior work [6, 10, 11], which operates in the logarithmic space of G/R and G/B [25]. We found that the R/G, B/G chromaticity histogram performs better, as it aligns with our model's output space. See the supplemental material for an ablation study.

In addition to the chromaticity histogram, $\mathbf{H}_c$, of the image colors, and building on prior work [6, 11], we augment it with the square-rooted chromaticity histogram of the image's edges, $\mathbf{H}_e$, where the image edges are computed as follows:

$$\mathbf{E}^{(x,y)} = \frac{1}{8} \sum_{\Delta x, \Delta y} \left| \mathbf{I}'^{(x,y)} - \mathbf{I}'^{(x+\Delta x, y+\Delta y)} \right|, \tag{7}$$

where $\mathbf{E}$ represents the image edges, $\mathbf{I}'$ refers to the raw image in 3D tensor form (height, width, channels), and $\Delta x, \Delta y \in \{-1, 0, 1\}$ with $(\Delta x, \Delta y) \neq (0, 0)$. The edge histogram, $\mathbf{H}_e$, is computed from $\mathbf{E}$ using Eqs. 5 and 6.

Our histogram feature is constructed by concatenating these two histograms, $\mathbf{H}_c$ and $\mathbf{H}_e$. Since this histogram feature is first processed by convolutional (conv) layers, as shown in Fig. 3, we follow [6] by appending additional channels that encode the positional information of the u/v coordinates in histogram space, which helps capture spatial relationships within the histogram feature. Consequently, our final histogram feature, $\mathbf{H} \in \mathbb{R}^{h \times h \times 4}$, consists of 2 channels representing the chromaticity of the image and its edges, along with the additional u/v coordinate channels.

This feature is processed through a series of conv layers with ELU activation [21]. The resulting latent representation undergoes adaptive average pooling before passing through a linear layer to produce the histogram's latent feature vector, $\mathbf{v}_h \in \mathbb{R}^{16}$, as follows:

$$\mathbf{v}_h = f_h(\mathbf{H}), \tag{8}$$

where f_h denotes the sub-model (i.e., conv layers, ELU activation, pooling, and linear layer) that maps the histogram feature into its latent space, as shown in Fig. 3.

2.3. Illuminant estimation

Both feature vectors, $\mathbf{v}_t$ and $\mathbf{v}_h$, are concatenated to produce the latent vector $\mathbf{v}$ and processed by an illuminant estimation sub-model as follows:

$$\mathbf{v} = \left[\mathbf{v}_t; \mathbf{v}_h\right], \tag{9}$$

$$\ell_c = f_\ell(\mathbf{v}), \tag{10}$$

where f_ℓ consists of a set of linear layers, with batch normalization applied to the first layer, followed by activation functions—except for the final layer, which outputs ℓ_c, the chromaticity vector of the scene illuminant. This vector is then transformed into an unnormalized RGB illuminant color by mapping $[\text{R/G, B/G}]^T \rightarrow [\text{R/G, 1, B/G}]^T$, followed by normalization via division by its L2 norm to produce the final illuminant color. We optimize f_t, f_h, and f_ℓ to minimize the angular error [36] between the predicted RGB illuminant color and the ground-truth RGB illuminant color.

3. Dataset

To train and validate our method, we require a dataset that includes contextual information (i.e., timestamp and geolocation) h h image. Existing datasets (e.g., [17, 19, 23, 28, 43, 44]) lack this essential information, motivating us to collect a new dataset using a smartphone camera that provides contextual metadata for each image. Specifically, we captured 3,224 linear raw images with the Samsung S24 Ultra's main camera, covering a wide range of scenes both indoors and outdoors, at various times of day (e.g., sunset, sunrise, noon, night). Our dataset includes images captured under various light sources (e.g., sunlight, incandescent, LED), as well as non-standard illuminant colors (e.g., colored LED light), which are not present in existing datasets [61]; see Fig. 4. Additionally, our dataset captures scenes under different weather conditions (sunny, cloudy, rain, snow, etc.). Example scenes can be found in the supplemental material.

We follow prior work [7, 43] in collecting ground-truth illuminant colors for each scene. Specifically, for each scene, we first capture an image with a calibration color chart, which is used to extract the illuminant color from the

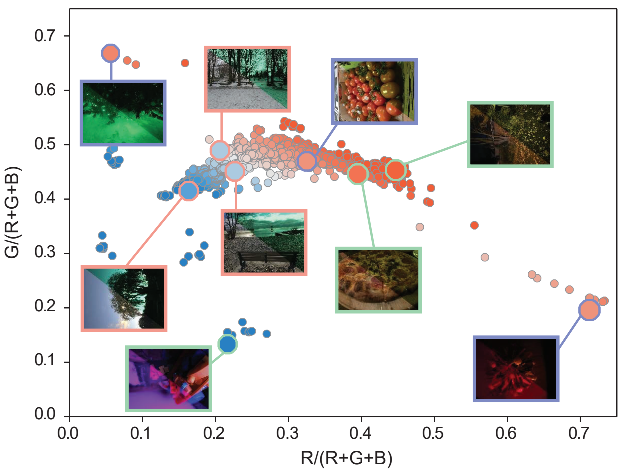

Figure 4. A plot of the rg chromaticity distribution of ground-truth illuminant colors for neutral white balance in our dataset. Example images in raw and sRGB spaces are shown (raw images are gamma-corrected for visualization). See supplemental material the rg chromaticity distribution of user-preference illuminants.

gray patches. Next, we capture an image of the same scene without the color chart (see Fig. 5), resulting in $\sim$6K images. After obtaining the ground-truth illuminant color, we discard all color chart images, leaving us with 3,224 images in our dataset. This approach allows us to test with natural images that mimic real-world scenarios, without the need to mask out the color chart patch.

Since our dataset includes a wide variety of lighting conditions, such as sunset/sunrise, night scenes, and artificial light, we generate a "user-preference" ground truth in addition to the neutral ground truth obtained from the color chart for each scene in our dataset. This is to account for human incomplete chromatic adaptation in such scenes [50, 60] and user preference [20, 24]. Specifically, an expert photographer was asked to assign a ground truth illuminant to each scene to make it appear more natural, reflect real-world observations, and enhance the aesthetics of the image. Notably, the same person who captured the scene also performed the annotation, ensuring that the user-preference selection was based on real-world observations of how the scene should appear.

We validated the user-preference ground truth through a study with 20 participants, who selected the most natural image from pairs of white-balanced images corresponding to the user-preference ground truth and the neutral ground truth (obtained from the color chart). The user-preference ground truth was selected in 71.95% of the trials, confirming its preference. See the supplemental material for additional details. In our experiments, both ground-truth types—neutral and user-preference—were used for training and evaluation (Sec. 4).

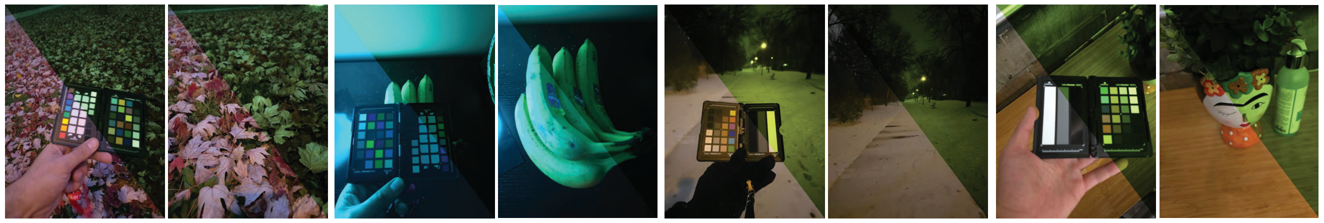
Figure 5. For each scene in our dataset, an image with a color chart placed in the scene was captured. The gray patches on the color chart were used to measure the ground-truth illuminant. These color-chart images were discarded, and only the images of the scenes without the color chart were used for the training, validation, and testing sets. For each example, we show both raw and sRGB images, with the raw images gamma-corrected to enhance visualization.

Additionally, we created a mask for regions illuminated by non-dominant illuminants in each scene. These masks ensure that all scenes have only one dominant illuminant, matching the color of the neutral ground truth, without confounding effects from other illuminants. We applied these masks to the training images when training our method and others. To preserve privacy, sensitive information, such as car plates and faces, has been blurred.

In addition to raw images, both ground-truth illuminant types, and contextual and capture information, we provide additional valuable auxiliary data to broaden the impact of our dataset. This includes locally tone-mapped sRGB images rendered by an expert, that can serve as ground-truth for applications beyond white-balance correction, such as neural ISPs [35, 38, 39, 65]. More information about the dataset can be found in the supplemental material. We organize our dataset into 2,619 raw images for training, 205 raw images for validation, and 400 raw images for testing.

4. Experiments

Implementation details: We train the model using the Adam optimizer [40], for 400 epochs with betas set to (0.9, 0.999) and a weight decay factor of 10^{-9}. A warm-up strategy is applied to gradually increase the learning rate from 10^{-6} to 10^{-3} over the first 5 epochs. After this, we use a cosine annealing schedule [48]. Following [6], we employ a batch-size increment strategy during training, starting with a batch size of 8 and doubling it every 100 epochs. Following prior work [6, 7], we use images of size 384×256 pixels in all experiments, which is a reasonable size for camera pipelines to reduce computational overhead. For our method, we use histograms with $h = 48$ bins. The histogram boundaries are determined by computing the 10th and 95th percentiles of the chromaticity values along each chromaticity axis from the training set. We report the results of our method both with and without the inclusion of noise stats, $\mathbf{n}$, and SNR stats, $\mathbf{r}$.

Results: We report the results of our method alongside several others benchmarked on the proposed dataset. Our method is compared with various statistical-based approaches [12, 14, 15, 18, 19, 26, 32, 56, 57, 62], camera-specific learning-based methods [4, 5, 11, 16, 29, 30, 37, 41, 42, 53, 59, 66, 67], and camera-independent techniques [2, 6, 13]. For the camera-specific learning-based approaches, including our method, all models were trained on our training set.

For the camera-independent methods [2, 6], we present results from models trained on the NUS [19] and Cube++ [23] datasets. Additionally, we report results from fine-tuned versions of these models [2, 6], incorporating our proposed dataset. We also evaluate camera-specific (CS) models of these methods [2, 6], which were trained exclusively on our training data, without any additional datasets. For further details, refer to the supplemental material.

Table 1 shows the results on the testing set of our dataset. We report the mean, median, best 25%, worst 25%, worst 5%, tri-mean, and maximum angular errors between the estimated illuminant colors and the ground truth colors for each method. Results are provided for both neutral and user-preference ground truth, with two models, one trained for each ground-truth type, except for the models SIIE [2], SIIE (tuned) [2], C5 [6], and C5 (tuned) [6], which were trained on images from the NUS [19] and Cube++ [23] datasets that do not include user-preference ground truth.

Additionally, we report the total number of parameters for methods that involve tunable or learnable parameters. The results in Table 1 are based on images without masking regions illuminated by a light source other than the dominant one used to obtain the ground truth, reflecting real-world scenarios where scenes with a single light source are not always guaranteed. For completeness, additional results are provided in the supplementary material, where regions illuminated by light sources different from the ground-truth illuminant color are masked out.

To further demonstrate the effectiveness of our method on existing DSLR datasets, where contextual metadata is typically unavailable, we report our results on the Simple Cube++ dataset [23]. Although geolocation metadata is absent, the dataset provides capture settings such as ISO and exposure time from the DSLR camera ISP.

Table 1. **Results on the testing set**. We report the mean, median, best 25%, worst 25%, tri-mean, and maximum angular errors for each method on neutral and user-preference white-balance ground-truth illuminants, presented in the format (neutral / user-preference). Symbols $\mathbf{n}$ and $\mathbf{r}$ represent noise stats and SNR stats, respectively. The best and second-best results are highlighted.

Method	**Mean**	**Med.**	**Best 25%**	**Worst 25%**	**Worst 5%**	**Tri.**	**Max**	**#params (K)**
GW [15]	6.23 / 5.54	6.01 / 4.52	1.02 / 1.00	12.43 / 11.91	18.05 / 19.81	5.76 / 4.65	28.09 / 31.70	-
SoG [26]	4.45 / 3.59	3.54 / 2.17	0.74 / 0.67	9.61 / 8.81	14.94 / 15.61	3.77 / 2.62	23.22 / 29.36	-
GE-1st [62]	4.21 / 3.33	3.29 / 2.12	0.71 / 0.55	9.34 / 8.29	15.34 / 15.19	3.49 / 2.36	27.81 / 28.80	-
GE-2nd [62]	4.11 / 3.18	3.17 / 1.89	0.70 / 0.58	9.09 / 7.79	14.90 / 14.85	3.35 / 2.18	25.09 / 29.42	-
Max-RGB [14]	4.01 / 2.61	2.92 / 1.88	1.06 / 0.97	8.57 / 5.69	14.01 / 11.07	3.30 / 1.94	34.22 / 23.63	-
wGE [32]	3.96 / 3.07	2.96 / 1.91	0.62 / 0.49	8.95 / 7.79	14.94 / 15.08	3.21 / 2.10	31.33 / 30.54	-
PCA [19]	4.42 / 3.63	3.54 / 1.95	0.70 / 0.61	9.59 / 9.19	14.46 / 16.16	3.76 / 2.52	22.87 / 31.58	-
MSGP [57]	6.39 / 5.69	5.72 / 4.38	0.94 / 0.98	13.39 / 12.73	20.97 / 22.33	5.64 / 4.68	37.25 / 36.92	-
GI [56]	4.70 / 4.93	3.19 / 2.97	0.44 / 0.78	11.63 / 12.22	20.48 / 21.35	3.48 / 3.53	36.34 / 36.02	-
TECC [12]	4.12 / 3.17	3.23 / 1.91	0.74 / 0.55	9.10 / 7.83	14.73 / 14.48	3.46 / 2.17	27.08 / 28.87	-
Gamut (pixels) [30]	3.77 / 2.40	2.81 / 1.49	0.77 / 0.63	8.31 / 5.84	12.93 / 11.18	3.16 / 1.63	21.53 / 23.31	0.636
Gamut (edges) [30]	4.45 / 3.94	3.52 / 3.04	1.08 / 1.01	9.51 / 8.50	15.00 / 15.25	3.70 / 3.18	28.55 / 29.46	324
Gamut (1st) [30]	4.10 / 3.65	3.08 / 2.67	0.72 / 0.90	9.26 / 8.25	14.75 / 14.69	3.28 / 2.91	21.38 / 26.35	279
NIS [29]	4.58 / 3.90	3.76 / 2.57	0.77 / 0.75	9.81 / 9.10	14.70 / 15.66	3.87 / 2.97	20.76 / 31.80	0.078
Classification-CC [53]	2.73 / 1.61	1.98 / 1.16	0.61 / 0.36	6.03 / 3.60	9.37 / 6.07	2.18 / 1.27	19.34 / 9.53	58,384
FFCC [11]	2.62 / 1.50	1.46 / 0.81	0.37 / 0.24	6.89 / 3.99	16.59 / 8.43	1.66 / 0.95	48.97 / 18.60	12
FFCC (capture info) [11]	2.31 / 1.35	1.38 / 0.80	0.34 / 0.23	5.82 / 3.51	12.44 / 7.36	1.60 / 0.88	47.67 / 16.96	36.9
FC4 [37]	3.80 / 2.65	2.78 / 2.25	0.85 / 0.85	8.61 / 5.14	15.06 / 7.52	2.86 / 2.37	25.74 / 11.42	1,705
APAP (GW) [5]	3.74 / 2.09	3.14 / 1.67	0.93 / 0.49	7.72 / 4.44	11.09 / 6.89	3.26 / 1.76	16.43 / 9.02	0.289
SIIE [2]	4.09 / -	3.25 / -	0.91 / -	8.97 / -	15.96 / -	3.37 / -	43.24 / -	1,008
SIIE (tuned) [2]	3.15 / -	2.22 / -	0.51 / -	7.28 / -	12.14 / -	2.46 / -	34.52 / -	1,008
SIIE (tuned-CS) [2]	3.14 / 1.74	2.20 / 1.20	0.50 / 0.32	7.39 / 4.06	13.61 / 6.73	2.41 / 1.30	38.96 / 9.44	1,008
KNN (raw) [4]	2.44 / 1.41	1.51 / 0.83	0.36 / 0.20	6.13 / 3.66	11.64 / 7.01	1.66 / 0.93	28.95 / 13.49	757
Quasi-U-CC [13]	3.85 / 3.26	2.97 / 1.81	0.55 / 0.58	8.47 / 8.24	13.30 / 15.78	3.25 / 2.21	24.74 / 32.17	54,421
Quasi-U-CC (tuned) [13]	3.11 / 2.54	2.27 / 1.43	0.49 / 0.44	7.14 / 6.57	11.53 / 13.62	2.44 / 1.63	22.67 / 33.74	54,421
BoCF [41]	3.54 / 2.14	2.68 / 1.59	0.96 / 0.48	7.31 / 4.74	11.40 / 7.85	2.96 / 1.72	22.31 / 19.68	59
C4 [66]	1.92 / 1.49	1.30 / 0.90	0.36 / 0.24	4.64 / 3.82	9.08 / 7.53	1.40 / 1.03	21.55 / 18.09	5,116
CWCC [42]	3.65 / 2.30	2.71 / 1.72	0.82 / 0.67	7.96 / 4.95	12.52 / 9.65	2.99 / 1.81	18.66 / 20.81	101
C5 [6]	3.22 / -	2.51 / -	0.78 / -	6.97 / -	10.54 / -	2.68 / -	16.38 / -	412
C5 (tuned) [6]	1.91 / -	1.24 / -	0.38 / -	4.57 / -	8.43 / -	1.38 / -	17.22 / -	412
C5 (tuned-CS) [6]	1.95 / 1.25	1.32 / 0.84	0.37 / 0.23	4.72 / 2.94	8.15 / 4.98	1.44 / 0.93	16.78 / 9.23	172
TLCC [59]	2.71 / 2.74	2.06 / 1.83	0.66 / 0.69	5.89 / 6.30	10.30 / 12.90	2.17 / 1.99	21.44 / 33.33	32,910
PCC [67]	3.03 / 1.67	2.13 / 1.20	0.53 / 0.40	7.08 / 3.79	11.20 / 6.84	2.34 / 1.27	16.82 / 12.33	0.378
RGP [18]	4.59 / 4.56	3.13 / 2.81	0.43 / 0.70	11.13 / 11.27	18.79 / 20.25	3.53 / 3.31	32.11 / 33.98	-
CFCC [16]	3.07 / 1.54	2.20 / 1.05	0.73 / 0.39	6.87 / 3.55	12.55 / 6.98	2.36 / 1.13	23.34 / 14.40	0.283
Ours (w/o $\mathbf{n}$, w/o $\mathbf{r}$)	1.93 / 1.26	1.35 / 0.77	0.38 / 0.23	4.56 / 3.13	9.48 / 5.88	1.43 / 0.88	22.63 / 15.90	4.83
Ours (w/o $\mathbf{n}$, w/ $\mathbf{r}$)	1.89 / 1.23	1.18 / 0.79	0.32 / 0.24	4.74 / 3.01	10.10 / 5.07	1.30 / 0.90	24.99 / 12.91	4.93
Ours (w/ $\mathbf{n}$, w/o $\mathbf{r}$)	1.87 / 1.20	1.24 / 0.72	0.37 / 0.24	4.58 / 2.99	9.82 / 5.58	1.30 / 0.82	29.46 / 15.12	4.93
Ours (w/ $\mathbf{n}$, w/ $\mathbf{r}$)	1.84 / 1.20	1.24 / 0.77	0.35 / 0.19	4.41 / 2.95	9.17 / 5.12	1.32 / 0.87	35.42 / 12.71	5.03

We trained our model using the histogram feature $\mathbf{H}$, along with capture features including ISO (i), exposure time (e), and SNR stats ($\mathbf{r}$), deliberately excluding the contextual metadata that is not available in the Simple Cube++ DSLR dataset. The results of our method, both with and without the SNR stats, are compared to other methods in Table 2. As shown, our method outperforms competing methods across most evaluation metrics.

Inference time: As shown in Table 1, our method performs comparably to or outperforms prior methods, while maintaining a compact model with only approximately 5K parameters. This lightweight design results in faster runtimes compared to other methods that achieve competitive results, namely C4 [66] and tuned C5 [6].

Table 2. **Results on the testing set of the 'Simple Cube++' dataset [23]**. We report the mean, median, best 25%, worst 25%, tri-mean, and maximum angular errors for each method. Symbols $\mathbf{c}$, i, e, and $\mathbf{r}$ refer to the time-capture feature, ISO, exposure time, and SNR stats, respectively. The best and second-best results are highlighted.

Method	**Mean**	**Med.**	**Best 25%**	**Worst 25%**	**Worst 5%**	**Tri.**	**Max**
FFCC [11]	1.38	0.58	0.18	4.05	10.30	0.68	45.20
FC4 [37]	3.72	2.02	0.47	9.93	17.18	2.40	29.92
C4 [66]	1.16	0.65	0.24	3.00	6.72	0.73	13.95
C5 (tuned-CS) [6]	1.19	0.60	0.18	3.26	7.61	0.68	15.62
TLCC [59]	1.82	1.15	0.39	4.49	9.63	1.20	19.02
Ours ($\mathbf{c} = [i, e]^T$)	1.08	0.59	0.17	2.84	5.53	0.69	11.03
Ours ($\mathbf{c} = [i, e, \mathbf{r}^T]^T$)	1.01	0.60	0.17	2.68	5.71	0.65	10.89

Table 3. Processing time on an AMD Ryzen Threadripper PRO 3975WX CPU460 and an NVIDIA RTX A6000 GPU.

Method	CPU (ms)	GPU (ms)	FLOPs
C4 [66]	49.03	11.28	2.28G
C5 [6]	8.28	4.50	103.54M
Ours	0.55	0.45	16.78M

In Table 3, we present the processing times of our model, C4, and C5 on an AMD Ryzen Threadripper PRO 3975WX CPU and an NVIDIA RTX A6000 GPU. Our fast runtime performance is particularly well-suited for mobile camera ISPs, where limited computational latency is crucial due to the processing demands of other modules (e.g., denoising, local tone mapping) per frame. Our model runs in just 0.25 ms on the DSP and 0.80 ms on the CPU of the Samsung S24 Ultra.

Ablation studies: We conducted ablation studies to evaluate the impact of each input feature on the validation set of our dataset. Specifically, we assessed our method by excluding either the time-capture feature ($\mathbf{c}$) or the histogram feature ($\mathbf{H}$). Additionally, we tested the method using only the time feature ($\mathbf{p}$) and noise stats ($\mathbf{n}$), while excluding the histogram feature ($\mathbf{H}$). We also evaluated the method using the histogram feature along with the time feature and noise stats, excluding other capture information. Furthermore, we examined the accuracy of our method using all input features except for $\mathbf{p}$. Lastly, we present the results when all input features are used, but without the noise stats ($\mathbf{n}$) and the SNR stats ($\mathbf{r}$). These results are presented in Table 4.

As shown in Table 4, using only the time feature ($\mathbf{p}$) and noise stats ($\mathbf{n}$) yields a reasonable accuracy (2.37° mean angular error), compared to FFCC [11], which achieves 2.19° mean angular error on the validation set. This validates the usefulness of time-of-day and noise information in providing contextual clues. As expected, incorporating the histogram feature ($\mathbf{H}$), which represents scene colors, along with the time-of-day and noise stats, significantly boosts accuracy, as demonstrated by the results in the fourth row. We further investigate the impact of using noise information in the last three rows: using SNR stats ($\mathbf{r}$), using noise stats ($\mathbf{n}$), or both. The combination of both noise features yields the lowest mean angular error. Additional ablation studies are provided in the supplementary material.

5. Conclusion and limitations

We presented a method for in-camera AWB correction that leverages contextual information. Specifically, we proposed a lightweight model that leverages contextual metadata (notably time-of-day derived from timestamp and geolocation) to guide the illuminant estimation process. In addition to this contextual metadata, we incorporate image colors in the form of histogram features, as well as capture information such as ISO, shutter speed, and noise stats, to help

Table 4. **Results of ablation studies on the validation set** with neutral white-balance ground truth. We report the mean, median, best 25%, and worst 25% angular errors for our method with various configurations. Symbols $\mathbf{H}$, $\mathbf{p}$, $\mathbf{n}$, and $\mathbf{r}$ denote the histogram feature, time feature, noise stats, and SNR stats, respectively. The symbol $\mathbf{m}$ refers to the capture information feature, which includes ISO (i), shutter speed (s), and flash status (f), all of which are used in our input time-capture feature $\mathbf{c}$. The best results are highlighted.

H	m	p	n	r	Mean	Med.	Best 25%	Worst 25%	Worst 5%
✓	✗	✗	✗	✗	2.03	1.36	0.31	4.90	8.90
✗	✓	✓	✓	✓	2.29	1.81	0.50	4.95	8.58
✗	✗	✓	✓	✗	2.37	1.59	0.45	5.48	9.03
✓	✗	✓	✓	✗	1.75	1.17	0.29	4.13	7.26
✓	✓	✗	✓	✓	1.89	1.21	0.31	4.60	9.41
✓	✓	✓	✗	✗	1.85	1.32	0.35	4.26	7.53
✓	✓	✓	✗	✓	1.72	1.16	0.30	4.13	8.16
✓	✓	✓	✓	✗	1.67	1.07	0.29	4.04	7.90
✓	✓	✓	✓	✓	1.66	1.20	0.33	3.77	6.95

the model distinguish between artificial and natural scenes and improve its final accuracy. Our method is fast and can achieve high frame rates on modern smartphone DSPs and CPUs, while maintaining accurate illuminant color estimation.

A key contribution of this work is a large-scale dataset of raw images captured by a consumer smartphone camera, along with the necessary contextual metadata for training and evaluating our method. Beyond the traditional neutral white-balance ground truth extracted from a calibration color chart placed in each scene, we also include a user-preference ground truth that targets the observer's preference, validated through a user study. Results based on both ground truth types demonstrate that our method achieves comparable or superior performance to existing methods, which require larger models.

While our method represents a promising solution for mobile camera ISPs, its dependency on contextual metadata limits its optimal accuracy to devices that provide geolocation data, which may not be available on DSLR cameras. Additionally, while the contextual metadata is device-independent (i.e., the differences across devices are expected to be minimal), our method relies on additional capture information (i.e., ISO, shutter speed, noise, and SNR stats) and image colors, making it inherently camera-dependent in design. This dependency prevents our trained model from generalizing to new devices without fine-tuning or re-training. However, this issue can be mitigated through color calibration (e.g., [46]), which can be extended to calibrate capture information as well—by performing a pre-processing mapping from the new camera space (for both color and capture information) to the camera space used during training. Additional discussion on cross-camera generalization is provided in the supplementary material.

References

[1] Abdelrahman Abdelhamed, Abhijith Punnappurath, and Michael S Brown. Leveraging the availability of two cameras for illuminant estimation. In *CVPR*, 2021. 2

[2] Mahmoud Afifi and Michael S Brown. Sensor-independent illumination estimation for DNN models. In *BMVC*, 2019. 1, 6, 7

[3] Mahmoud Afifi and Michael S Brown. Deep white-balance editing. In *CVPR*, 2020. 1

[4] Mahmoud Afifi, Brian Price, Scott Cohen, and Michael S Brown. When color constancy goes wrong: Correcting improperly white-balanced images. In *CVPR*, 2019. 6, 7

[5] Mahmoud Afifi, Abhijith Punnappurath, Graham Finlayson, and Michael S Brown. As-projective-as-possible bias correction for illumination estimation algorithms. *JOSA A*, 36 (1):71–78, 2019. 6, 7

[6] Mahmoud Afifi, Jonathan T Barron, Chloe LeGendre, Yun-Ta Tsai, and Francois Bleibel. Cross-camera convolutional color constancy. In *ICCV*, 2021. 1, 2, 4, 6, 7, 8

[7] Mahmoud Afifi, Zhenhua Hu, and Liang Liang. Optimizing illuminant estimation in dual-exposure HDR imaging. In *ECCV*, 2024. 1, 2, 5, 6

[8] Vivek Agarwal, Besma R Abidi, Andreas Koschan, and Mongi A Abidi. An overview of color constancy algorithms. *Journal of Pattern Recognition Research*, 1(1):42–54, 2006. 1

[9] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012. 4

[10] Jonathan T Barron. Convolutional color constancy. In *ICCV*, 2015. 1, 2, 4

[11] Jonathan T Barron and Yun-Ta Tsai. Fast Fourier color constancy. In *CVPR*, 2017. 2, 4, 6, 7, 8

[12] Simone Bianco and Marco Buzzelli. Truncated edge-based color constancy. In *ICCE*, 2022. 6, 7

[13] Simone Bianco and Claudio Cusano. Quasi-unsupervised color constancy. In *CVPR*, 2019. 1, 6, 7

[14] David H Brainard and Brian A Wandell. Analysis of the Retinex theory of color vision. *Journal of the Optical Society of America A*, 3(10):1651–1661, 1986. 6, 7

[15] Gershon Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin Institute*, 310(1): 1–26, 1980. 1, 6, 7

[16] Marco Buzzelli and Simone Bianco. A convolutional framework for color constancy. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2024. 6, 7

[17] Cheng Cheng, Kai-Fu Yang, Xue-Mei Wan, Leanne Lai Hang Chan, and Yong-Jie Li. Nighttime color constancy using robust gray pixels. *JOSA A*, 41(3):476–488, 2024. 5

[18] Cheng Cheng, Kai Fu Yang, Xue Mei Wan, Leanne Lai Hang Chan, and Yonge Jie Li. Nighttime color constancy using robust gray pixels. *JOSA A*, 41(3):476–488, 2024. 6, 7

[19] 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. *JOSA A*, 31(5):1049–1058, 2014. 2, 5, 6, 7

[20] Kyungah Choi and Hyeon-Jeong Suk. User-preferred color temperature adjustment for smartphone display under varying illuminants. *Optical Engineering*, 53(6):061708–061708, 2014. 5

[21] Djork-Arné Clevert. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015. 5

[22] 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. 3

[23] Egor Ershov, Alexey Savchik, Illya Semenkov, Nikola Banić, Alexander Belokopytov, Daria Senshina, Karlo Koščević, Marko Subašić, and Sven Lončarić. The cube++ illumination estimation dataset. *IEEE Access*, 8:227511–227527, 2020. 2, 5, 6, 7

[24] Egor Ershov, Alex Savchik, Denis Shepelev, Nikola Banić, Michael S Brown, Radu Timofte, Karlo Koščević, Michael Freeman, Vasily Tesalin, Dmitry Bocharov, et al. NTIRE 2022 challenge on night photography rendering. In *CVPRW*, 2022. 5

[25] Graham D Finlayson and Steven D Hordley. Color constancy at a pixel. *JOSA A*, 18(2):253–264, 2001. 4

[26] Graham D Finlayson and Elisabetta Trezzi. Shades of gray and colour constancy. In *CIC*, 2004. 1, 6, 7

[27] Karl R Gegenfurtner, David Weiss, and Marina Bloj. Color constancy in real-world settings. *Journal of Vision*, 24(2): 12–12, 2024. 1

[28] Peter Vincent Gehler, Carsten Rother, Andrew Blake, Tom Minka, and Toby Sharp. Bayesian color constancy revisited. In *CVPR*, 2008. 2, 5

[29] Arjan Gijsenij and Theo Gevers. Color constancy using natural image statistics and scene semantics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4): 687–698, 2011. 6, 7

[30] Arjan Gijsenij, Theo Gevers, and Joost Van De Weijer. Generalized gamut mapping using image derivative structures for color constancy. *International Journal of Computer Vision*, 86(2):127–139, 2010. 2, 6, 7

[31] Arjan Gijsenij, Theo Gevers, and Joost Van De Weijer. Computational color constancy: Survey and experiments. *IEEE Transactions on Image Processing*, 20(9):2475–2489, 2011. 1

[32] Arjan Gijsenij, Theo Gevers, and Joost Van De Weijer. Improving color constancy by photometric edge weighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):918–929, 2011. 1, 6, 7

[33] Thorsten Hansen, Sebastian Walter, and Karl R Gegenfurtner. Effects of spatial and temporal context on color categories and color constancy. *Journal of Vision*, 7(4):2–2, 2007. 1

[34] Samuel W Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Transactions on Graphics*, 35(6):1–12, 2016. 3

[35] Xuanhua He, Tao Hu, Guoli Wang, Zejin Wang, Run Wang, Qian Zhang, Keyu Yan, Ziyi Chen, Rui Li, Chengjun Xie, et al. Enhancing RAW-to-sRGB with decoupled style structure in Fourier domain. In *AAAI*, 2024. 6

[36] Steven D Hordley and Graham D Finlayson. Re-evaluating colour constancy algorithms. In *ICPR*, 2004. 5

[37] Yuanming Hu, Baoyuan Wang, and Stephen Lin. FC4: Fully convolutional color constancy with confidence-weighted pooling. In *CVPR*, 2017. 6, 7

[38] Andrey Ignatov, Anastasia Sycheva, Radu Timofte, Yu Tseng, Yu-Syuan Xu, Po-Hsiang Yu, Cheng-Ming Chiang, Hsien-Kai Kuo, Min-Hung Chen, Chia-Ming Cheng, et al. MicroISP: processing 32MP photos on mobile devices with deep learning. In *ECCV*, 2022. 6

[39] Woohyeok Kim, Geonu Kim, Junyong Lee, Seungyong Lee, Seung-Hwan Baek, and Sunghyun Cho. ParamISP: Learned forward and inverse ISPs using camera parameters. In *CVPR*, 2024. 6

[40] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[41] Firas Laakom, Nikolaos Passalis, Jenni Raitoharju, Jarno Nikkanen, Anastasios Tefas, Alexandros Iosifidis, and Moncef Gabbouj. Bag of color features for color constancy. *IEEE Transactions on Image Processing*, 29:7722–7734, 2020. 1, 6, 7

[42] Firas Laakom, Jenni Raitoharju, Jarno Nikkanen, Alexandros Iosifidis, and Moncef Gabbouj. Robust channel-wise illumination estimation. In *BMVC*, 2021. 2, 6, 7

[43] Firas Laakom, Jenni Raitoharju, Jarno Nikkanen, Alexandros Iosifidis, and Moncef Gabbouj. Intel-TAU: A color constancy dataset. *IEEE Access*, 9:39560–39567, 2021. 2, 5

[44] Shuwei Li and Robby T Tan. NightCC: Nighttime color constancy via adaptive channel masking. In *CVPR*, 2024. 5

[45] Orly Liba, Kiran Murthy, Yun-Ta Tsai, Tim Brooks, Tianfan Xue, Nikhil Karnad, Qiurui He, Jonathan T Barron, Dillon Sharlet, Ryan Geiss, et al. Handheld mobile photography in very low light. *ACM Transactions on Graphics*, 38(6):164–1, 2019. 3

[46] Yi-Tun Lin, Bianjiang Yang, Hao Xie, Wenbin Wang, Honghong Peng, and Jun Hu. Color constancy: How to deal with camera bias? In *BMVC*, 2023. 8

[47] Ce Liu, William T Freeman, Richard Szeliski, and Sing Bing Kang. Noise estimation from a single image. In *CVPR*, 2006. 4

[48] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6

[49] Zhongyu Lou, Theo Gevers, Ninghang Hu, Marcel P Lucassen, et al. Color constancy by deep learning. In *BMVC*, 2015. 1

[50] Ming Luo. A review of chromatic adaptation transforms. *Review of Progress in Coloration and Related Topics*, 30:77 – 92, 2008. 5

[51] Laurence T Maloney. Illuminant estimation as cue combination. *Journal of Vision*, 2(6):6–6, 2002. 1

[52] Jean H Meeus. *Astronomical algorithms*. Willmann-Bell, Incorporated, 1991. 3

[53] Seoung Wug Oh and Seon Joo Kim. Approaching the computational color constancy as a classification problem through deep learning. *Pattern Recognition*, 61:405–416, 2017. 1, 6, 7

[54] Varad A Pimpalkhute, Rutvik Page, Ashwin Kothari, Kishor M Bhurchandi, and Vipin Milind Kamble. Digital image noise estimation using DWT coefficients. *IEEE Transactions on Image Processing*, 30:1962–1972, 2021. 4

[55] Stanislav Pyatykh, Jürgen Hesser, and Lei Zheng. Image noise level estimation by principal component analysis. *IEEE Transactions on Image Processing*, 22(2):687–699, 2012. 4

[56] Yanlin Qian, Joni-Kristian Kamarainen, Jarno Nikkanen, and Jiri Matas. On finding gray pixels. In *CVPR*, 2019. 1, 6, 7

[57] Yanlin Qian, Said Pertuz, Jarno Nikkanen, Joni-Kristian Kämäräinen, and Jiri Matas. Revisiting gray pixel for statistical illumination estimation. In *VISSAP*. 2019. 1, 6, 7

[58] Wu Shi, Chen Change Loy, and Xiaoou Tang. Deep specialized network for illuminant estimation. In *ECCV*, pages 371–387, 2016. 1

[59] Yuxiang Tang, Xuejing Kang, Chunxiao Li, Zhaowen Lin, and Anlong Ming. Transfer learning for color constancy via statistic perspective. In *AAAI*, 2022. 2, 6, 7

[60] Shoji Tominaga, Takahiko Horiuchi, Shiori Nakajima, and Mariko Yano. Prediction of incomplete chromatic adaptation under illuminant A from images. In *CIC*, 2014. 2, 5

[61] Oguzhan Ulucan, Diclehan Ulucan, and Marc Ebner. Color constancy beyond standard illuminants. In *ICIP*, 2022. 2, 5

[62] Joost Van De Weijer, Theo Gevers, and Arjan Gijsenij. Edge-based color constancy. *IEEE Transactions on Image Processing*, 16(9):2207–2214, 2007. 1, 6, 7

[63] TC Van Flandern and KF Pulkkinen. Low-precision formulae for planetary positions. *Astrophysical Journal Supplement Series*, 41:391–411, 1979. 3

[64] Robert Walraven. Calculating the position of the sun. *Solar energy*, 20(5):393–397, 1978. 3

[65] Yazhou Xing, Zian Qian, and Qifeng Chen. Invertible image signal processing. In *CVPR*, 2021. 6

[66] Huanglin Yu, Ke Chen, Kaiqi Wang, Yanlin Qian, Zhaoxiang Zhang, and Kui Jia. Cascading convolutional color constancy. In *AAAI*, 2020. 1, 6, 7, 8

[67] Shuwei Yue and Minchen Wei. Color constancy from a pure color view. *Journal of the Optical Society of America A*, 40 (3):602–610, 2023. 6, 7

[68] Yi Zhang, Hongwei Qin, Xiaogang Wang, and Hongsheng Li. Rethinking noise synthesis and modeling in raw denoising. In *ICCV*, 2021. 4