Fast Fourier Color Constancy
Supplement

Jonathan T. Barron
barron@google.com

Yun-Ta Tsai
yuntatsai@google.com

1. Pretraining

In the paper we described the data term for our loss function $f(\cdot)$ which takes a toroidal PDF $P(i,j)$, fits a bivariate von Mises distribution to P, and then computes the negative log-likelihood of the true white point L^* under that distribution. This loss is non-convex, and therefore may behave erratically in the earliest training iterations. This issue is compounded by our differentiable BVM fitting procedure, which may be inacurate when P has a low concentration, which is often the case in early iterations. For this reason, we train our model in two stages: In the "pretraining" stage we replace the data term in our loss function with a more simple loss: straightforward logistic regression with respect to P and some ground-truth PDF P^* (Eq. 2), and then in the second training stage we use the data term described in the paper while using the output of pretraining to initialize the model. Because our regularization is also convex, using this pretraining loss makes our entire optimization problem convex and therefore straightforward to optimize, and (when coupled with our use of LBFGS for optimization instead of some SGD-like approach) also makes training deterministic.

Computing a logistic loss is straightforward: we compute a ground-truth PDF P^* from the ground-truth illuminant L^*, and then compute a standard logistic loss.

$$P^*(i,j) = \mathrm{mod}\left(\frac{L_u^* - u_{lo}}{h} - i, n\right) < 1 \quad (1)$$
$$\wedge\, \mathrm{mod}\left(\frac{L_v^* - v_{lo}}{h} - j, n\right) < 1$$

$$f_{\mathrm{pretrain}}(P) = -\sum_{i,j} P^*(i,j)\log(P(i,j)) \quad (2)$$

This loss behaves very similarly to the loss used in CCC [1], but it has the added benefit of being convex.

2. Backpropagation

The bivariate von Mises estimation procedure described in the paper can be thought of as a "layer" in a deep learning architecture, as our end-to-end training procedure requires that we be able to backpropagate through the fitting procedure and the loss computation. Here we present the gradients of the critical equations described in the paper.

$$\nabla_{\boldsymbol{\mu}} f(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -2\boldsymbol{\Sigma}^{-1}\left(\begin{bmatrix}L_u^*\\ L_v^*\end{bmatrix} - \boldsymbol{\mu}\right) \quad (3)$$

$$\nabla_{\boldsymbol{\Sigma}} f(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\left(\begin{bmatrix}L_u^*\\ L_v^*\end{bmatrix} - \boldsymbol{\mu}\right)\left(\begin{bmatrix}L_u^*\\ L_v^*\end{bmatrix} - \boldsymbol{\mu}\right)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}$$

$$\nabla_{P(i,j)}\boldsymbol{\mu} = \left(\frac{nh}{2\pi}\right)\begin{bmatrix}\frac{x_i\sin(\theta(i)) - y_i\cos(\theta(i))}{x_i^2+y_i^2}\\ \frac{x_j\sin(\theta(j)) - y_j\cos(\theta(j))}{x_j^2+y_j^2}\end{bmatrix} \quad (4)$$

$$\nabla_{P(i,j)}\boldsymbol{\Sigma} = h^2\begin{bmatrix}\bar{i}\left(\bar{i} - 2\,\mathrm{E}\left[\bar{i}\right]\right), & \left(\bar{i} - \mathrm{E}\left[\bar{i}\right]\right)\left(\bar{j} - \mathrm{E}\left[\bar{j}\right]\right) - \mathrm{E}\left[\bar{i}\right]\mathrm{E}\left[\bar{j}\right]\\ \left(\bar{i} - \mathrm{E}\left[\bar{i}\right]\right)\left(\bar{j} - \mathrm{E}\left[\bar{j}\right]\right) - \mathrm{E}\left[\bar{i}\right]\mathrm{E}\left[\bar{j}\right], & \bar{j}\left(\bar{j} - 2\,\mathrm{E}\left[\bar{j}\right]\right)\end{bmatrix}$$

By chaining these gradients together we can backpropagate the gradient of the loss back onto the input PDF P. Backpropagating through the softmax operation and the convolution (and illumination gain/bias) is straightforward and so is not detailed here.

3. Deep Models

In the main paper we stated that Models N, O, and P use an alternative parametrization to incorporate external features during training and testing. This parameterization allows our model to reason about things other than simple pixel and edge log-chroma histograms, like semantics and camera metadata. In the basic model presented in the main paper, we learn a set of weights $(\{F_k\}, G, B)$, where these weights determine the shape of the filters used during convolution and the per-color gain/bias applied to the output of that convolution. Let us abstractly refer to the concatenation of these (preconditioned, Fourier-domain) weights as $\mathbf{w}$, and let the loss contributed by the data term for training data instance i be $f_i(\mathbf{w})$ (here $f_i(\mathbf{w})$ does not just apply a loss, but first undoes the preconditioning transformation and maps from our real FFT vector space to a complex 2D FFT). Training our basic model can be thought of as simply finding

$$\arg\min_{\mathbf{w}} \sum_i f_i(\mathbf{w}) \quad (5)$$

To generalize our model, instead of learning a single model $\mathbf{w}$, we instead define a feature vector for each training instance $\mathbf{x}_i$ and learn a mapping from each $\mathbf{x}_i$ to some $\mathbf{w}_i$ such that the loss for all $\{\mathbf{w}_i\}$ is minimized. Instead of learning a single $\mathbf{w}$, we learn the weights in a small 2-layer neural network with a ReLU activation function, where those network weights define the mapping from features to FFCC parameters. The resulting optimization problem during training is:

$$\underset{\mathbf{W}_1,\mathbf{b}_1,\mathbf{W}_2,\mathbf{b}_2}{\arg\min} \sum_i f_i\left(\mathbf{W}_2 \max(0, \mathbf{W}_1\mathbf{x}_i + \mathbf{b}_1) + \mathbf{b}_2\right) \quad (6)$$

Like in all other experiments we train using batch L-BFGS, but instead of the two-stage training used in the shallow model (a convex "pretraining" loss and a nonconvex final loss), we have only one training stage: 64 iterations of LBFGS, in which we minimize a weighted sum of the two losses. Our input vectors $\{\mathbf{x}_i\}$ are whitened before training, and the whitening transformation is absorbed into $\mathbf{W}_1$ and $\mathbf{b}_1$ after training so that unwhitened features can be used at test-time. Our weights are initialized to random Gaussian noise, unlike the shallow model which is initialized to all zeros. Unlike our "shallow" model, in which $\mathbf{w}$ is regularized during training, for our "deep" models we do not directly regularize each $\mathbf{w}_i$ but instead indirectly regularize all $\mathbf{w}_i$ by minimizing the squared 2-norm of each $\mathbf{W}_i$ and $\mathbf{b}_i$. This use of weight decay to regularize our model depends critically on the frequency-domain preconditioning we use, which causes a simple weight decay to indirectly impose the careful smoothness regularizer that was constructed for our shallow model. Note that our "deep" model is equivalent to our "shallow" model if the input vector is empty (ie, $\mathbf{x}_i = [\,]$), as $\mathbf{b}_2$ would behave equivalently to $\mathbf{w}$ in that case. We use 4 hidden units for Models N and O, and 8 hidden units for Model P (which uses the concatenated features from both Models N and O). The magnitude of the noise used for initialization and of the weight decay for each layer of the network are tuned using cross-validation.

To produce the "metadata" features used in Models O and P we use the EXIF tags included in the Gehler-Shi dataset. Using external information in this way is unusual in the color constancy literature, which is why this aspect of our model is relegated to just two experiments (all figures and other results do not use external metadata). In contrast, camera manufacturers spend significant effort considering sensor spectral properties and other sources of information that may be useful when building a white balance system. For example, knowing that two images came from two different sensors (as is the case in the Gehler-Shi dataset) allows for a more careful treatment of absolute color and black body radiation. And knowing the absolute brightness of the scene (indicated by the camera's exposure time, etc) can be a useful cue for distinguishing between the bright light of the sun and the relatively low light of man made light sources. As the improved performance of Model O demonstrates, this other information is indeed informative and can induce a significant reduction in error. We use a compact feature vector that encodes the outer product of the exposure settings of the camera and the name of the camera sensor itself, all extracted from the EXIF tags included in the public dataset:

$$\begin{aligned}\mathbf{x}_i = &\mathrm{vec}(\\ &[\log(\mathrm{shutter\_speed}_i); \log(\mathrm{f\_number}_i); 1] \\ &\times [\mathbf{1}_{\mathrm{Canon1D}}(\mathrm{camera}_i), \mathbf{1}_{\mathrm{Canon5D}}(\mathrm{camera}_i), 1])\end{aligned} \quad (7)$$

Note that the Gehler-Shi dataset uses images from two different Canon cameras, as reflected here. The log of the shutter speed and F number are chosen as features because, in theory, their difference should be proportional to the log of the exposure value of the image, which should indicate the amount of light receiving by the camera sensor.

The "semantics" features used in Models N and P are simply the output of the CNN model used in [6], which was run on the pre-whitebalance image after it is center-cropped to a square and resized to 256×256. Because this image is in the sensor colorspace, before passing it to the CNN we scale the green channel by 0.6, apply a CCM, and apply an sRGB gamma curve. These semantic features have a modest positive effect.

4. Real Bijective FFT

In the paper we describe $\mathcal{F}_{\mathrm{v}}(Z)$, a FFT variant that takes the 2D FFT of a $n \times n$ real-valued 2D image Z and then linearizes it into a real-valued vector with no redundant values. Having this FFT-like one-to-one mapping between real 2D images and real 1D vectors enables our frequency-domain preconditioner.

Our modified FFT function is defined as:

$$\mathcal{F}_{\mathrm{v}}(Z) = \begin{bmatrix} \mathrm{Re}(\mathcal{F}(Z)(0 : n/2, 0)) \\ \mathrm{Re}(\mathcal{F}(Z)(0 : n/2, n/2)) \\ \mathrm{Re}(\mathcal{F}(Z)(0 : (n-1), 1 : (n/2 - 1))) \\ \mathrm{Im}(\mathcal{F}(Z)(1 : (n/2 - 1), 0)) \\ \mathrm{Im}(\mathcal{F}(Z)(1 : (n/2 - 1), n/2 - 1)) \\ \mathrm{Im}(\mathcal{F}(Z)(0 : (n-1), 1 : (n/2 - 1))) \end{bmatrix} \quad (8)$$

Where $\mathcal{F}(Z)(i, j)$ is the complex number at the zero-indexed (i, j) position in the FFT of Z, and $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ extract real and imaginary components, respectively. The output of $\mathcal{F}_{\mathrm{v}}(Z)$ is an n^2-dimensional vector, as it must be for our mapping to preserve all FFT coefficients with no redundancy. To preserve the scale of the FFT through this mapping we scale $\mathcal{F}_{\mathrm{v}}(Z)$ by $\sqrt{2}$, ignoring the entries that correspond to:

$$\mathrm{Re}(\mathcal{F}(Z)(0, 0))$$

$$\begin{aligned}&\mathrm{Re}(\mathcal{F}(Z)(0, n/2))\\&\mathrm{Re}(\mathcal{F}(Z)(n/2, 0))\\&\mathrm{Re}(\mathcal{F}(Z)(n/2, n/2))\end{aligned} \tag{9}$$

This scaling ensure that the magnitude of Z is preserved:

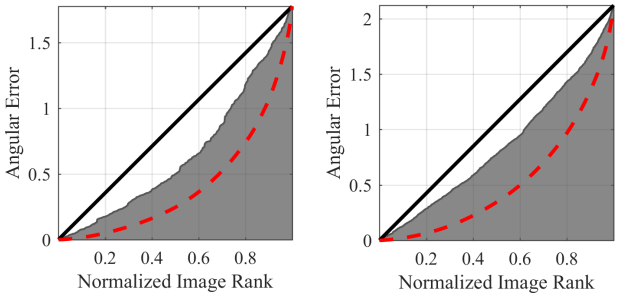
$$\|\mathcal{F}_{\mathrm{v}}(Z)\|^2 = |\mathcal{F}(Z)|^2 \tag{10}$$

To compute the inverse of $\mathcal{F}_{\mathrm{v}}(\cdot)$ we undo this scaling, undo the vectorization by filling in a subset of the elements of $\mathcal{F}(Z)$ from the vector representation, set the other elements of $\mathcal{F}(Z)$ such that Hermitian symmetry holds, and the invert the FFT.

5. Results

Because our model produces a complete posterior distribution over illuminants in the form of a covariance matrix $\mathbf{\Sigma}$, each of our illuminant estimates comes with a measure of confidence in the form of the entropy: $\frac{1}{2}\log|\mathbf{\Sigma}|$ (ignoring a constant shift). A low entropy suggests a tight concentration of the output distribution, which tends to be well-correlated with a low error. To demonstrate this we present a novel error metric, which is twice the area under the curve formed by ordering all images (the union of all test-set images from each cross-validation fold) by ascending entropy and normalizing by the number of images. In Figure 1 we visualize this error metric and show that our entropy-ordered error is substantially lower than the mean error for both of our datasets, which shows that a low entropy is suggestive of a low error. We are not aware of any other color constancy technique which explicitly predicts a confidence measure, and so we do not compare against any existing technique, but it can be demonstrated that if the entropy used to sort error is decorrelated with error (or, equivalently, if the error cannot be sorted due to the lack of the means to sort it) that entropy-ordered error will on average be equal to mean error.

To allow for a better understanding of our model's performance, we present images from the Gehler-Shi dataset [3, 5] (Figures 2-11) and the Canon 1Ds MkIII camera from the Cheng et al. dataset [2] (Figures 12-16). There results were produced using Model J presented in the main paper. For each dataset we perform three-fold cross validation, and with that we produce output predictions for each image along with an error measure (angular RGB error) and an entropy measure (the entropy of the covariance matrix of our predicted posterior distribution over illuminants). The images chosen here were selected by sorting images from each dataset by increasing error and evenly sampling images according to that ordering (10 from Gehler-Shi, 5 from the smaller Cheng dataset). This means that the first image in each sequence is the lowest error image, and the last is the highest. The rendered images include the color checker



(a) Gehler-Shi dataset [3, 5]		(b) Cheng et al. dataset [2]
EO Error:	1.287	1.696
Mean Error:	1.775	2.121

Figure 1: By sorting each image by the entropy of its posterior distribution we can show that entropy correlates with error. Here we sort the images by ascending entropy and plot the cumulative sum of the error, filling in the area under that curve with gray. If entropy was not correlated with error we would expect the area under the curve to match the black line, and if entropy was perfectly correlated with error then the area under the curve would exactly match the dashed red line. We report twice the area under the curve as "entropy-ordered" error (mean error happens to be twice the area under the diagonal line).

used in creating the ground-truth illuminants used during training, but it should be noted that these color checkers are masked out when these images are used during training and evaluation. For each image we present: a) the input image, b) the predicted bivariate von Mises distribution over illuminants, c) our estimated illuminant and white-balanced image (produced by dividing the estimated illuminant into the input image), and d) the ground-truth illuminant and white-balanced image. Our log-chroma histograms are visualized using gray light de-aliasing to assign each (i, j) coordinate a color, with a blue dot indicating the location/color of the ground-truth illuminant, a red dot indicating our predicted illuminant $\boldsymbol{\mu}$ and a red ellipse indicating the predicted covariance of the illuminant $\mathbf{\Sigma}$. The bright lines in the histogram indicate the locations where $u = 0$ or $v = 0$. The reported entropy of the covariance $\mathbf{\Sigma}$ corresponds to the spread of the covariance (low entropy = small spread). We see that our low error predictions tend to have lower entropies, and vice versa, confirming our analysis in Figure 1. We also see that the ground-truth illuminant tends to lie within the estimated covariance matrix, though not always for the largest-error images.

In Figure 17 we visualize a set of images taken from a Nexus 6 in the HDR+ mode [4] after being white-balanced by Model Q in the main paper (the version designed to run on thumbnail images).

6. Color Rendering

All images are rendered by applying the RGB gains implied by the estimated illuminant, applying some color correction matrix (CCM) and then applying an sRGB gamma-correction function (the C_{linear} to C_{srgb} mapping in `http://en.wikipedia.org/wiki/SRGB`). For each camera in the datasets we use we estimate our own CCMs using the imagery, which we present here. These CCMs do not affect our illuminant estimation or our results, and are only relevant to our visualizations. Each CCM is estimated through an iterative least-squares process in which we alternatingly: 1) estimate the ground-truth RGB gains for each image from a camera by solving a least-squares system using our current CCM, and 2) use our current gains to estimate a row-normalized CCM using a constrained least-squares solve. Our estimated ground-truth gains are not used in this paper. For the ground-truth sRGB colors of the Macbeth color chart we use the hex values provided here: `http://en.wikipedia.org/wiki/ColorChecker#Colors` which we linearize.

$$\text{GehlerShi, Canon1D} \begin{bmatrix} 2.2310 & -1.5926 & 0.3616 \\ -0.1494 & 1.4544 & -0.3050 \\ 0.1641 & -0.6588 & 1.4947 \end{bmatrix}$$

$$\text{GehlerShi, Canon5D} \begin{bmatrix} 1.7494 & -0.8470 & 0.0976 \\ -0.1565 & 1.4380 & -0.2815 \\ 0.0786 & -0.5070 & 1.4284 \end{bmatrix}$$

$$\text{Cheng, Canon1DsMkIII} \begin{bmatrix} 1.7247 & -0.7791 & 0.0544 \\ -0.1436 & 1.4632 & -0.3195 \\ 0.0589 & -0.4625 & 1.4037 \end{bmatrix}$$

$$\text{Cheng, Canon600D} \begin{bmatrix} 1.8988 & -0.9897 & 0.0909 \\ -0.2058 & 1.6396 & -0.4338 \\ 0.0749 & -0.7030 & 1.6281 \end{bmatrix}$$

$$\text{Cheng, FujifilmXM1} \begin{bmatrix} 1.4183 & -0.2497 & -0.1686 \\ -0.2230 & 1.6449 & -0.4219 \\ 0.0785 & -0.5980 & 1.5195 \end{bmatrix}$$

$$\text{Cheng, NikonD5200} \begin{bmatrix} 1.3792 & -0.3134 & -0.0659 \\ -0.0826 & 1.3759 & -0.2932 \\ 0.0483 & -0.4553 & 1.4070 \end{bmatrix}$$

$$\text{Cheng, OlympusEPL6} \begin{bmatrix} 1.6565 & -0.4971 & -0.1595 \\ -0.3335 & 1.7772 & -0.4437 \\ 0.0895 & -0.7023 & 1.6128 \end{bmatrix}$$

$$\text{Cheng, PanasonicGX1} \begin{bmatrix} 1.5629 & -0.5117 & -0.0512 \\ -0.2472 & 1.7590 & -0.5117 \\ 0.1395 & -0.8945 & 1.7550 \end{bmatrix}$$

$$\text{Cheng, SamsungNX2000} \begin{bmatrix} 1.5770 & -0.4351 & -0.1419 \\ -0.1747 & 1.5225 & -0.3477 \\ 0.0573 & -0.6397 & 1.5825 \end{bmatrix}$$

$$\text{Cheng, SonyA57} \begin{bmatrix} 1.5963 & -0.5545 & -0.0418 \\ -0.1343 & 1.5331 & -0.3988 \\ 0.0563 & -0.4026 & 1.3463 \end{bmatrix}$$

References

[1] J. T. Barron. Convolutional color constancy. *ICCV*, 2015.

[2] D. Cheng, D. K. Prasad, and M. S. Brown. Illuminant estimation for color constancy: why spatial-domain methods work and the role of the color distribution. *JOSA A*, 2014.

[3] P. Gehler, C. Rother, A. Blake, T. Minka, and T. Sharp. Bayesian color constancy revisited. *CVPR*, 2008.

[4] S. W. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. T. Barron, F. Kainz, J. Chen, and M. Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *SIGGRAPH Asia*, 2016.

[5] L. Shi and B. Funt. Re-processed version of the gehler color constancy dataset of 568 images. http://www.cs.sfu.ca/ colour/data/.

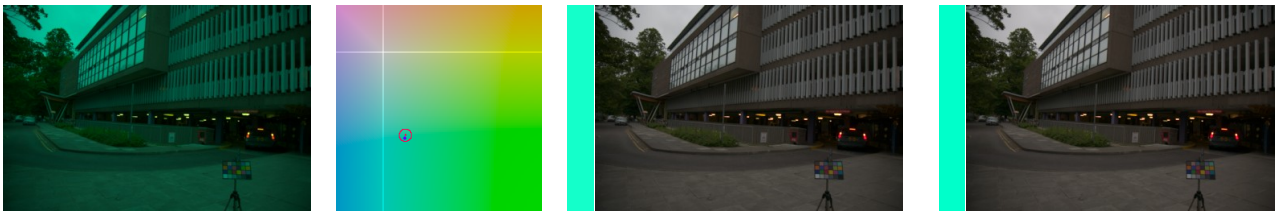
[6] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. *CVPR*, 2014.

(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 2: A result from the Gehler-Shi dataset using Model J. Error = $0.02°$, entropy = -6.48

(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 3: A result from the Gehler-Shi dataset using Model J. Error = $0.26°$, entropy = -6.55



(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

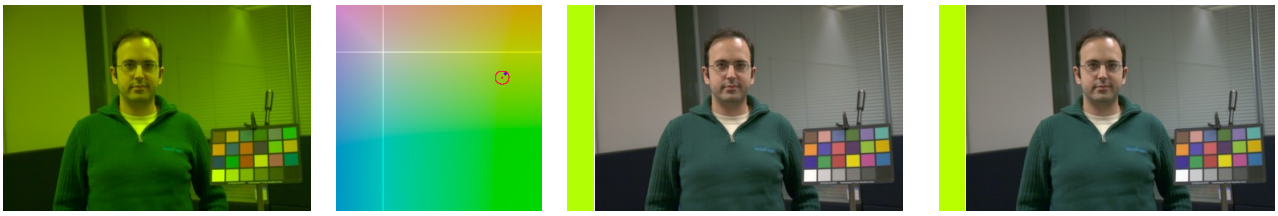
Figure 4: A result from the Gehler-Shi dataset using Model J. Error = $0.46°$, entropy = -6.91

(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 5: A result from the Gehler-Shi dataset using Model J. Error = $0.63°$, entropy = -6.37

(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 6: A result from the Gehler-Shi dataset using Model J. Error = $0.83°$, entropy = -6.62

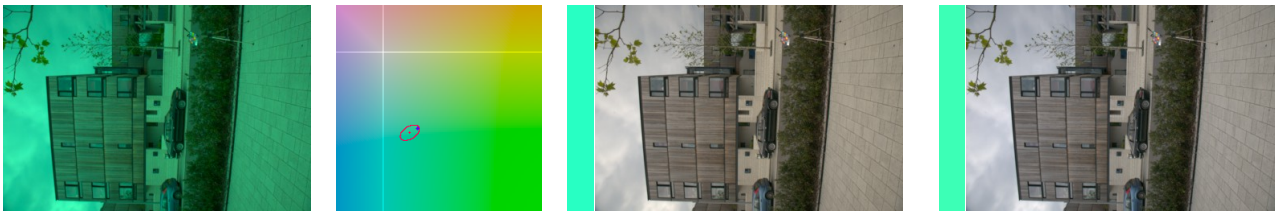


(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 7: A result from the Gehler-Shi dataset using Model J. Error = $1.19°$, entropy = -6.71

(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 8: A result from the Gehler-Shi dataset using Model J. Error = $1.61°$, entropy = -6.88

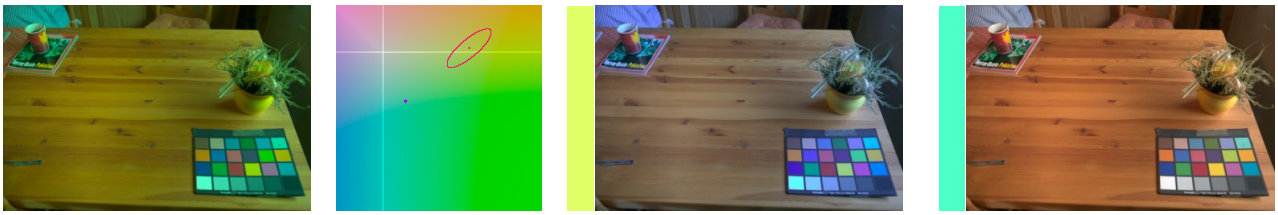


(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 9: A result from the Gehler-Shi dataset using Model J. Error = $2.35°$, entropy = -6.32

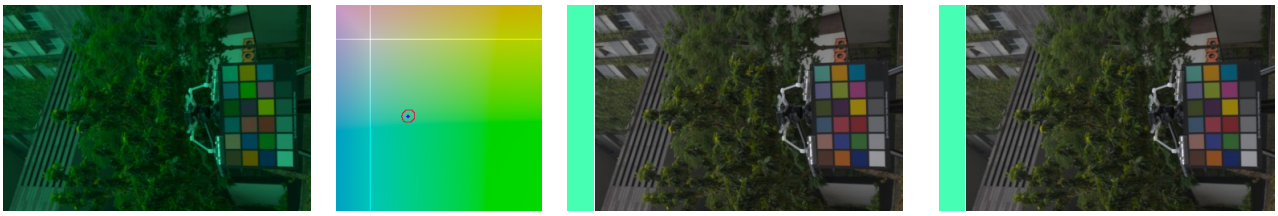
(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 10: A result from the Gehler-Shi dataset using Model J. Error = $3.84°$, entropy = -5.28



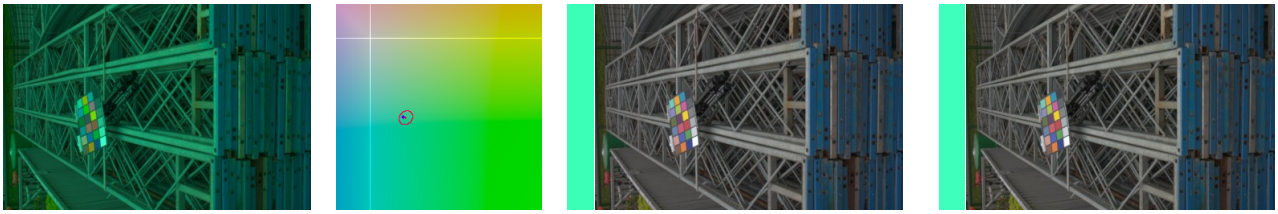
(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 11: A result from the Gehler-Shi dataset using Model J. Error = $21.64°$, entropy = -4.95



(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 12: A result from the Cheng dataset using Model J. Error = $0.12°$, entropy = -6.82



(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 13: A result from the Cheng dataset using Model J. Error = $0.64°$, entropy = -6.69

(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 14: A result from the Cheng dataset using Model J. Error = $1.37°$, entropy = -6.48

(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 15: A result from the Cheng dataset using Model J. Error = $2.69°$, entropy = -5.82

(a) Input Image (b) Illuminant Posterior (c) Our prediction (d) Ground Truth

Figure 16: A result from the Cheng dataset using Model J. Error = $17.85°$, entropy = -3.04

Figure 17: A sampling of unedited HDR+[4] images from a Nexus 6, after being processed with Model Q.