Convolutional Color Constancy
Supplemental Material

Jonathan T. Barron
barron@google.com

1. Exponential Decay SGD

Though we use conventional (batch) L-BFGS to finalize optimization when training our model, optimization can be sped up by using stochastic (non-batch) gradient descent techniques prior to L-BFGS. Inspired by recent advances in "second-order" stochastic gradient descent techniques [2, 6, 7], we developed a novel variant of stochastic gradient descent based on exponential decaying different quantities at different rates, which we found to work well on our task.

Pseudocode for our "exponential decay SGD" technique can be seen in Algorithm 1. This technique is similar to RMSProp [6] and AdaDelta [7], in that we maintain a moving average model of the gradient-squared, and then divide the gradient by the square-root of the average-gradient-squared before taking a gradient-descent step. But in addition to maintaining a moving average gradient-squared using exponential decay, we also maintain a moving average estimate of the gradient and of the loss. The moving average gradient serves as an alternative to commonly used "mini-batches", where instead of computing the average gradient of n datapoints before taking a gradient descent step, we compute n different gradients and take n different gradient descent steps while averaging in past gradient estimates to prevent dramatic jumps during optimization. This appears to help optimization, especially in our domain where our training set sizes are somewhat small.

Many SGD techniques use an adaptive learning rate, where the learning rate is increased every epoch if optimization succeeds (ie, the loss decreases) and the learning rate is decreased if optimization fails (ie, the loss is increasing or oscillating). To generalize this idea we maintain a moving average of the loss for the entire dataset and compare every sampled datapoint's loss to that average. If a datapoint's loss appears is less than the average we slightly increase the step size, otherwise we slightly decrease the step size. In contrast to a per-epoch learning rate revision, this approach allows the step size to vary quickly during just a single epoch of optimization, thereby speeding up optimization.

We parametrized our technique in terms of half-life rather than using decay multipliers, which makes these parameters easier to reason about. For example, we found if effective to set the half-life for the average loss to be roughly the size of the dataset, so that the average loss is always a reflection of the entire dataset. The half-life of the gradient we set to be small — about the size of a mini-batch, and the half-life of the gradient-squared we set to be significantly larger than that of the gradient but less than that of the loss.

To ensure that our running average estimates are correct even at the beginning of optimization, we model each moving average as the ratio of two quantities. Though we describe our algorithm as randomly sampling datapoints until convergence, in practice we optimize for a fixed number of epochs (in our experiments, 50) and for each epoch we randomly order our data and then sample every datapoint in that random order, thereby improving the coverage of our training data.

2. Additional Images

To provide a better understanding of our results we present additional images from the reprocessed [5] Color Checker Dataset [4]. For each image we show the input image produced by the camera and the ground-truth illumination and white-balanced image. All images are shown as cropped squares, for the sake of easy visualization. We also show the output of our model ("CCC"), and that of the three best-performing baseline techniques (two variants of the "Corrected Moment" algorithm [3] and the technique of Cheng et al.[1]. Though color checkers are visible in these images, the color checkers are cropped out of the images before being evaluated by any color constancy algorithm. To prevent "cherry picking", we sorted the 568 images in the Color Checker Dataset by the average errors of the four algorithms being evaluated (ordering the images from "easy" to "hard") and evenly sampled images 1, 50, ... 551.

References

[1] 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.

Algorithm 1 Exponential Decay SGD

```
Hyperparameters:
  $\beta_f = 10^3$          // The half-life of our exponentially decayed loss.
  $\beta_g = 10^1$          // The half-life of our exponentially decayed gradient.
  $\beta_h = 10^2$          // The half-life of our exponentially decayed gradient-squared.
  $\alpha = 10^{-3}$        // The initial step-size.
  $\alpha^+ = 1.0001$       // The amount to decrease the step-size when the loss decreases.
  $\alpha^- = 0.999$        // The amount to decrease the step-size when the loss increases.
  $\epsilon = 10^{-5}$      // A small constant to prevent divide-by-zero.
Input:
  $\theta$        // The initial model parameters.
  $\{X\}$         // The training dataset.
  $L(\cdot)$      // The loss function
  $\lambda_f \leftarrow 2^{-1/\beta_f}$     // Convert the loss half-life to a decay.
  $\lambda_g \leftarrow 2^{-1/\beta_g}$     // Convert the gradient half-life to a decay.
  $\lambda_h \leftarrow 2^{-1/\beta_h}$     // Convert the gradient-squared half-life to a decay.
  $f_n \leftarrow 0, f_d \leftarrow 0$     // Initialize the moving average loss.
  $g_n \leftarrow 0, g_d \leftarrow 0$     // Initialize the moving average gradient.
  $h_n \leftarrow 0, h_d \leftarrow 0$     // Initialize the moving average gradient-squared.
  while not converged do
    $X_t \sim \{X\}$     // Sample a datapoint.
    $(f_t, g_t) \leftarrow L(X_t, \theta)$     // Compute the loss and gradient.
    $f_n \leftarrow \lambda_f f_n + (1 - \lambda_f) f_t$     // Decay the moving average numerators and add the new quantities.
    $g_n \leftarrow \lambda_g g_n + (1 - \lambda_g) g_t$
    $h_n \leftarrow \lambda_h h_n + (1 - \lambda_h) g_t^2$
    $f_d \leftarrow \lambda_f f_d + (1 - \lambda_f)$     // Decay the moving average denominators and add the new mass.
    $g_d \leftarrow \lambda_g g_d + (1 - \lambda_g)$
    $h_d \leftarrow \lambda_h h_d + (1 - \lambda_f)$
    $\bar{f} = f_n / f_d$     // Estimate the moving average loss.
    $\bar{g} = g_n / g_d$     // Estimate the moving average gradient.
    $\bar{h} = h_n / h_d$     // Estimate the moving average gradient-squared.
    $\Delta f = f_t - \bar{f}$     // Compare the moving average loss to the current datapoint's loss.
    if $\Delta f \leq 0$ then     // If the loss appears to be decreasing....
      $\alpha \leftarrow \alpha^+ \times \alpha$     // then increase the step size...
    else
      $\alpha \leftarrow \alpha^- \times \alpha$     // otherwise decrease the step size.
    end if
    $\theta = \theta - \alpha \frac{\bar{g}}{\sqrt{\bar{h} + \epsilon^2}}$     // Update the model parameters.
  end while
  return return $\theta$
```

[2] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 2011.

[3] G. D. Finlayson. Corrected-moment illuminant estimation. *ICCV*, 2013.

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

[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] T. Tieleman and G. Hinton. Lecture 6.5- rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.

[7] M. D. Zeiler. Adadelta: An adaptive learning rate method. *CoRR*, 2012.

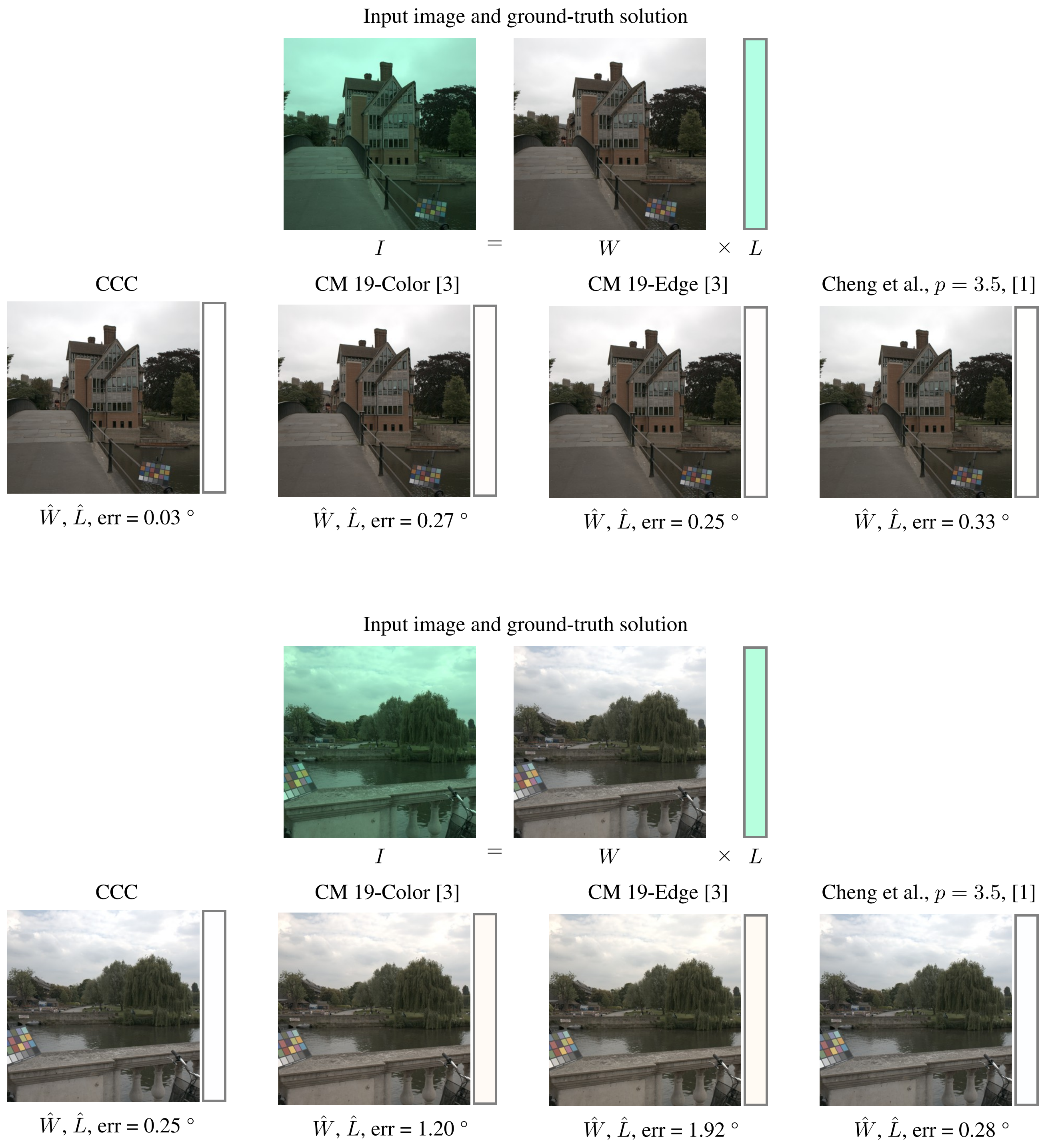
Input image and ground-truth solution
I = W × L
CCC
CM 19-Color [3]
CM 19-Edge [3]
Cheng et al., $p = 3.5$, [1]
$\hat{W}$, $\hat{L}$, err = 0.03 °
$\hat{W}$, $\hat{L}$, err = 0.27 °
$\hat{W}$, $\hat{L}$, err = 0.25 °
$\hat{W}$, $\hat{L}$, err = 0.33 °
Input image and ground-truth solution
I = W × L
CCC
CM 19-Color [3]
CM 19-Edge [3]
Cheng et al., $p = 3.5$, [1]
$\hat{W}$, $\hat{L}$, err = 0.25 °
$\hat{W}$, $\hat{L}$, err = 1.20 °
$\hat{W}$, $\hat{L}$, err = 1.92 °
$\hat{W}$, $\hat{L}$, err = 0.28 °

Figure 1: For each scene we present the input image produced by the camera alongside the ground-truth illumination color and white-balanced image. Images are shown in sRGB, normalized to the 98th percentile. For our algorithm and three baseline algorithms we show the estimated illumination and white-balanced image, as well as the error in degrees of the estimated illumination with respect to the ground-truth. Recovered illuminations are rendered with respect to ground-truth, such that "white" is correct, and any deviation from white is an error.

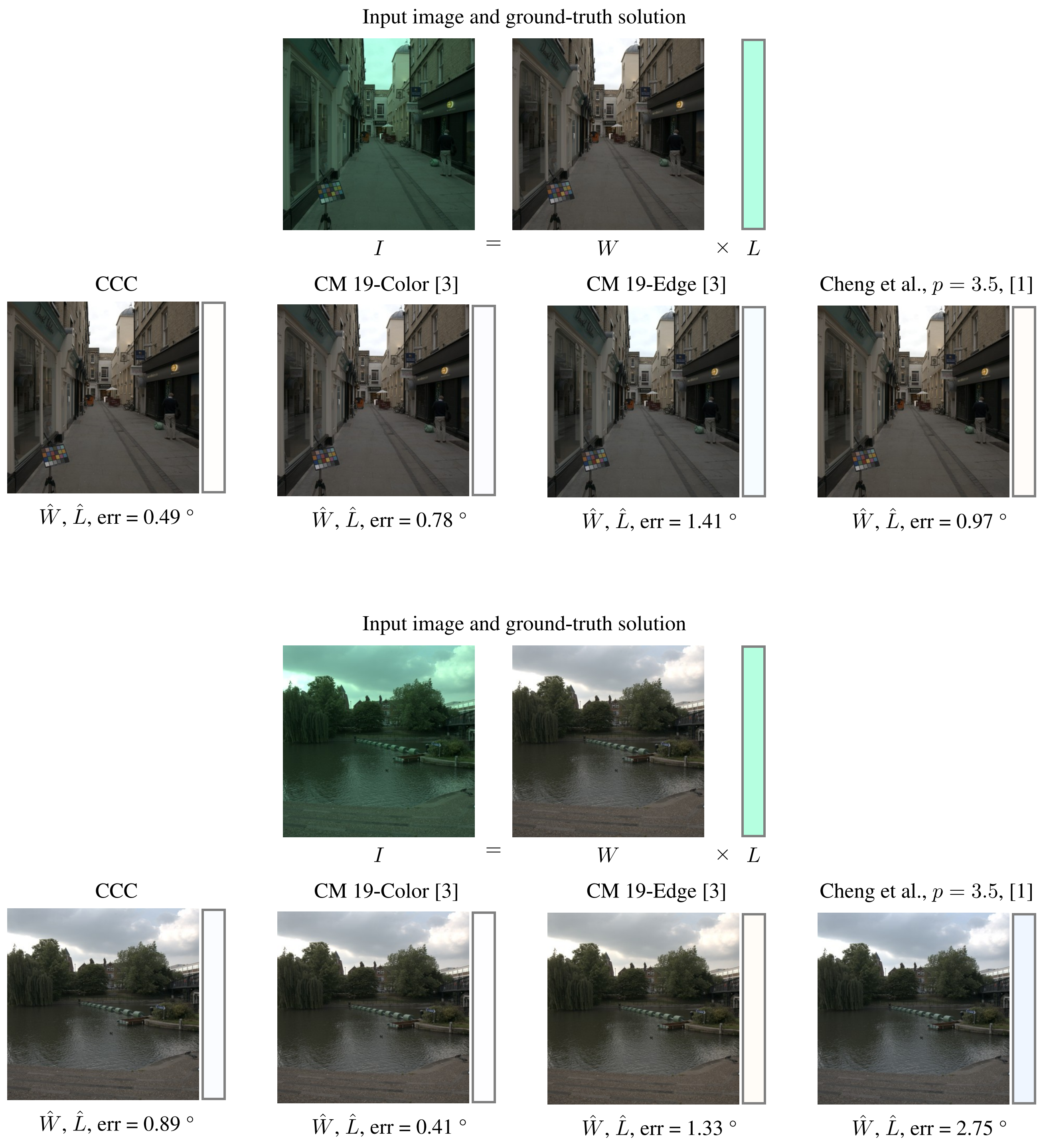
Input image and ground-truth solution
I = W × L
CCC
CM 19-Color [3]
CM 19-Edge [3]
Cheng et al., $p = 3.5$, [1]
$\hat{W}$, $\hat{L}$, err = 0.49 °
$\hat{W}$, $\hat{L}$, err = 0.78 °
$\hat{W}$, $\hat{L}$, err = 1.41 °
$\hat{W}$, $\hat{L}$, err = 0.97 °
Input image and ground-truth solution
I = W × L
CCC
CM 19-Color [3]
CM 19-Edge [3]
Cheng et al., $p = 3.5$, [1]
$\hat{W}$, $\hat{L}$, err = 0.89 °
$\hat{W}$, $\hat{L}$, err = 0.41 °
$\hat{W}$, $\hat{L}$, err = 1.33 °
$\hat{W}$, $\hat{L}$, err = 2.75 °

Figure 2: Additional results in the same format as Figure 1.

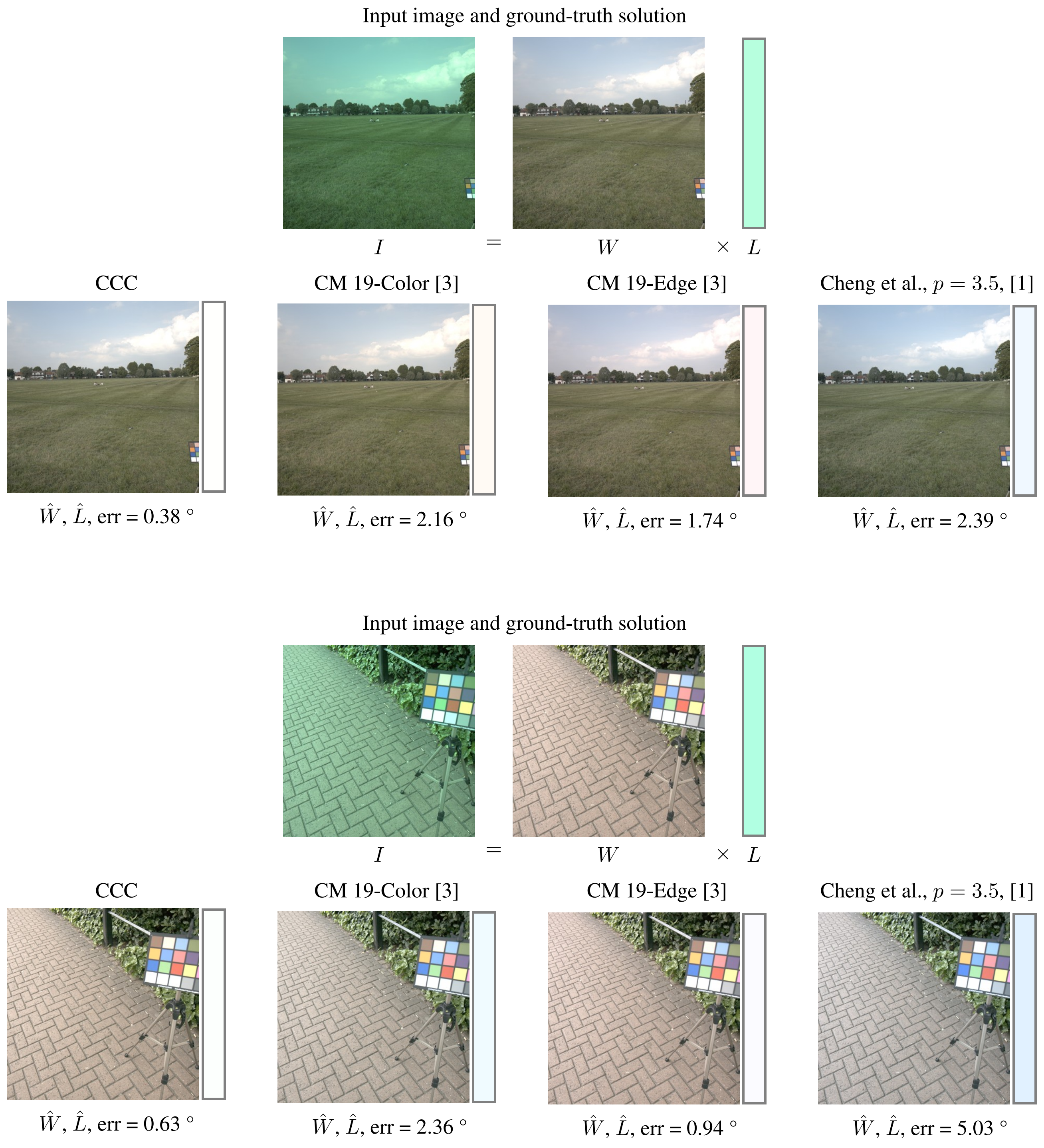
Input image and ground-truth solution
I = W × L
CCC
CM 19-Color [3]
CM 19-Edge [3]
Cheng et al., $p = 3.5$, [1]
$\hat{W}$, $\hat{L}$, err = 0.38 °
$\hat{W}$, $\hat{L}$, err = 2.16 °
$\hat{W}$, $\hat{L}$, err = 1.74 °
$\hat{W}$, $\hat{L}$, err = 2.39 °
Input image and ground-truth solution
I = W × L
CCC
CM 19-Color [3]
CM 19-Edge [3]
Cheng et al., $p = 3.5$, [1]
$\hat{W}$, $\hat{L}$, err = 0.63 °
$\hat{W}$, $\hat{L}$, err = 2.36 °
$\hat{W}$, $\hat{L}$, err = 0.94 °
$\hat{W}$, $\hat{L}$, err = 5.03 °

Figure 3: Additional results in the same format as Figure 1.

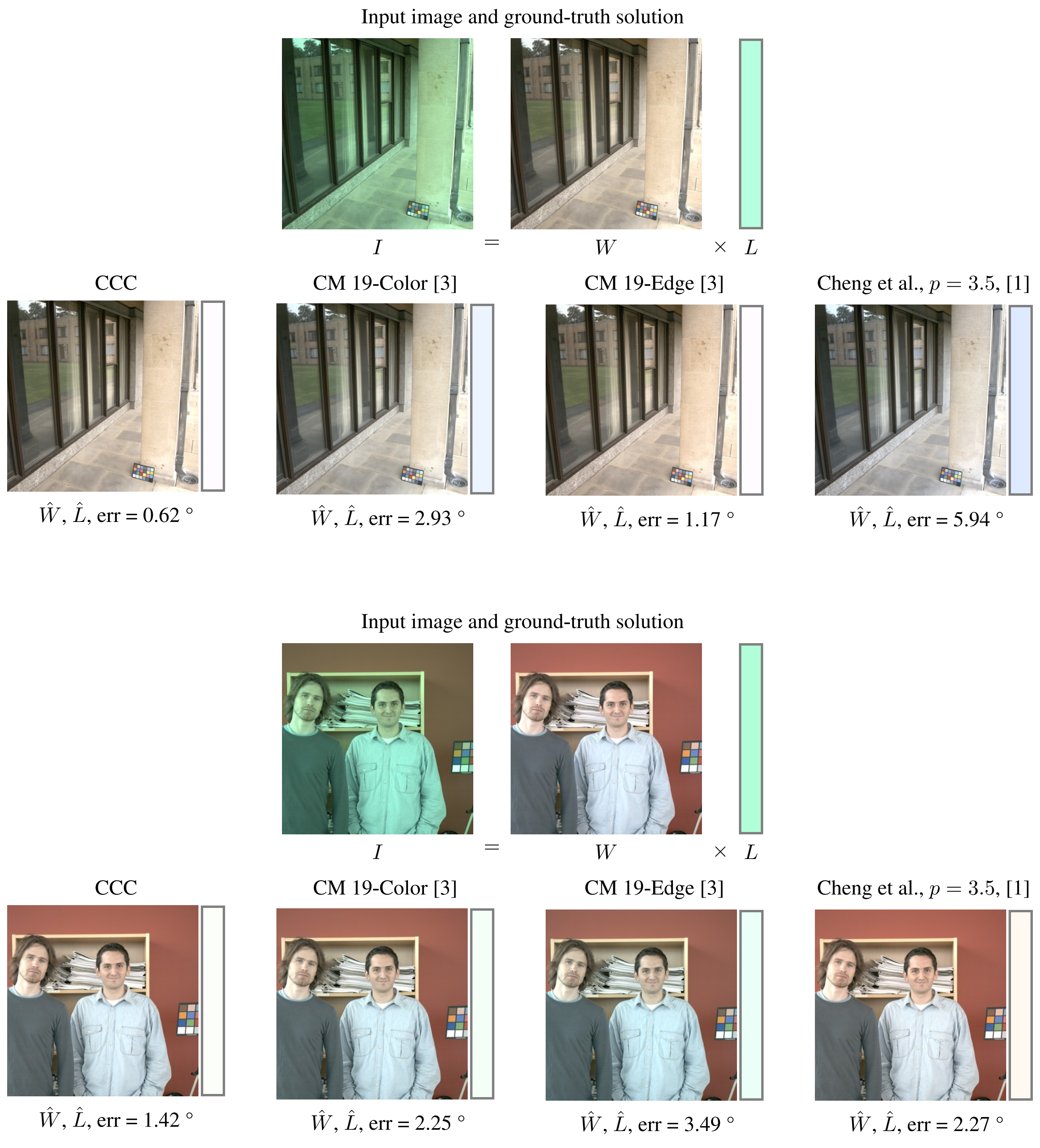

Figure 4: Additional results in the same format as Figure 1.

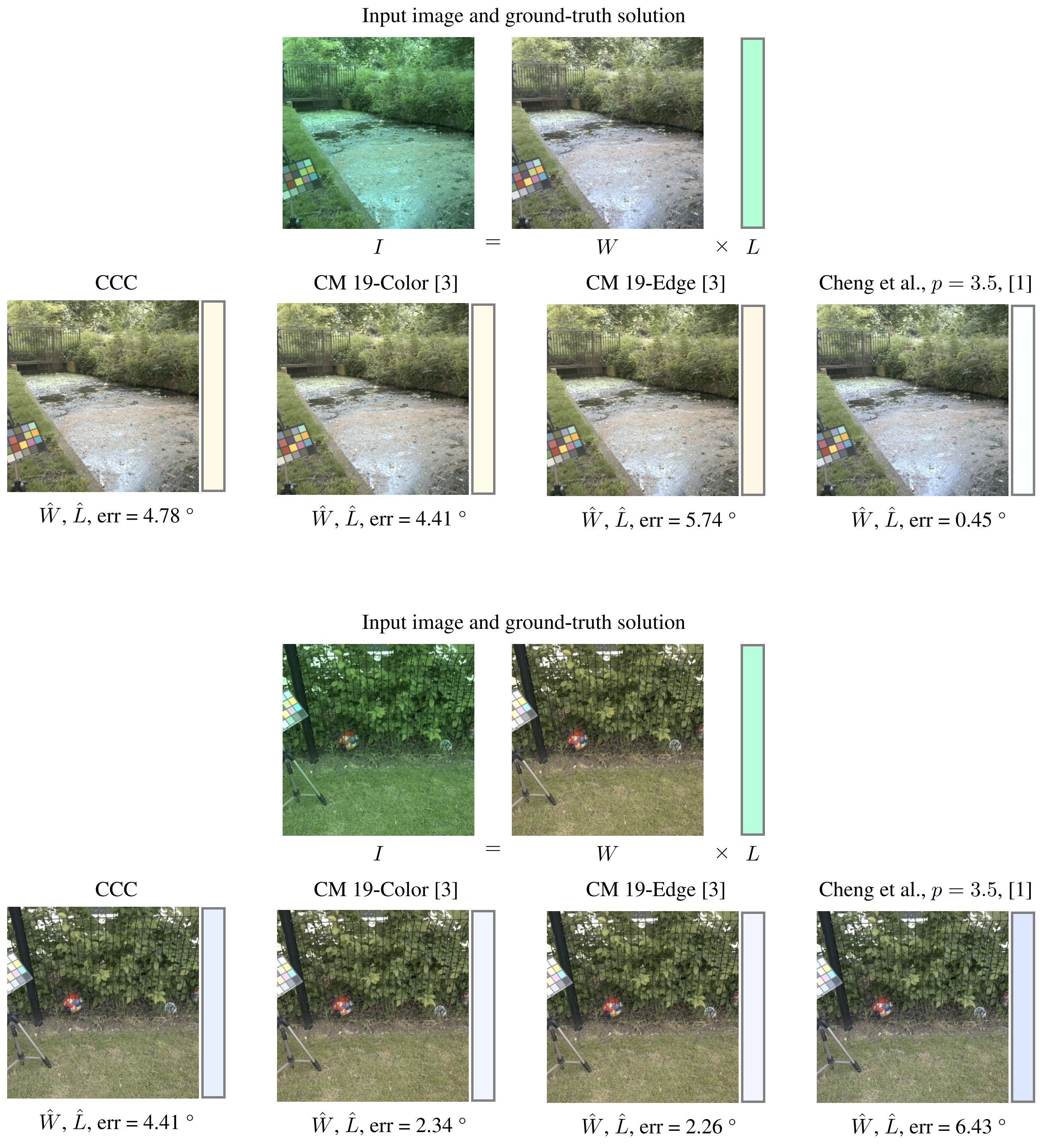
Input image and ground-truth solution
I = W × L
CCC
CM 19-Color [3]
CM 19-Edge [3]
Cheng et al., $p = 3.5$, [1]
$\hat{W}$, $\hat{L}$, err = 4.78 °
$\hat{W}$, $\hat{L}$, err = 4.41 °
$\hat{W}$, $\hat{L}$, err = 5.74 °
$\hat{W}$, $\hat{L}$, err = 0.45 °
Input image and ground-truth solution
I = W × L
CCC
CM 19-Color [3]
CM 19-Edge [3]
Cheng et al., $p = 3.5$, [1]
$\hat{W}$, $\hat{L}$, err = 4.41 °
$\hat{W}$, $\hat{L}$, err = 2.34 °
$\hat{W}$, $\hat{L}$, err = 2.26 °
$\hat{W}$, $\hat{L}$, err = 6.43 °

Figure 5: Additional results in the same format as Figure 1.

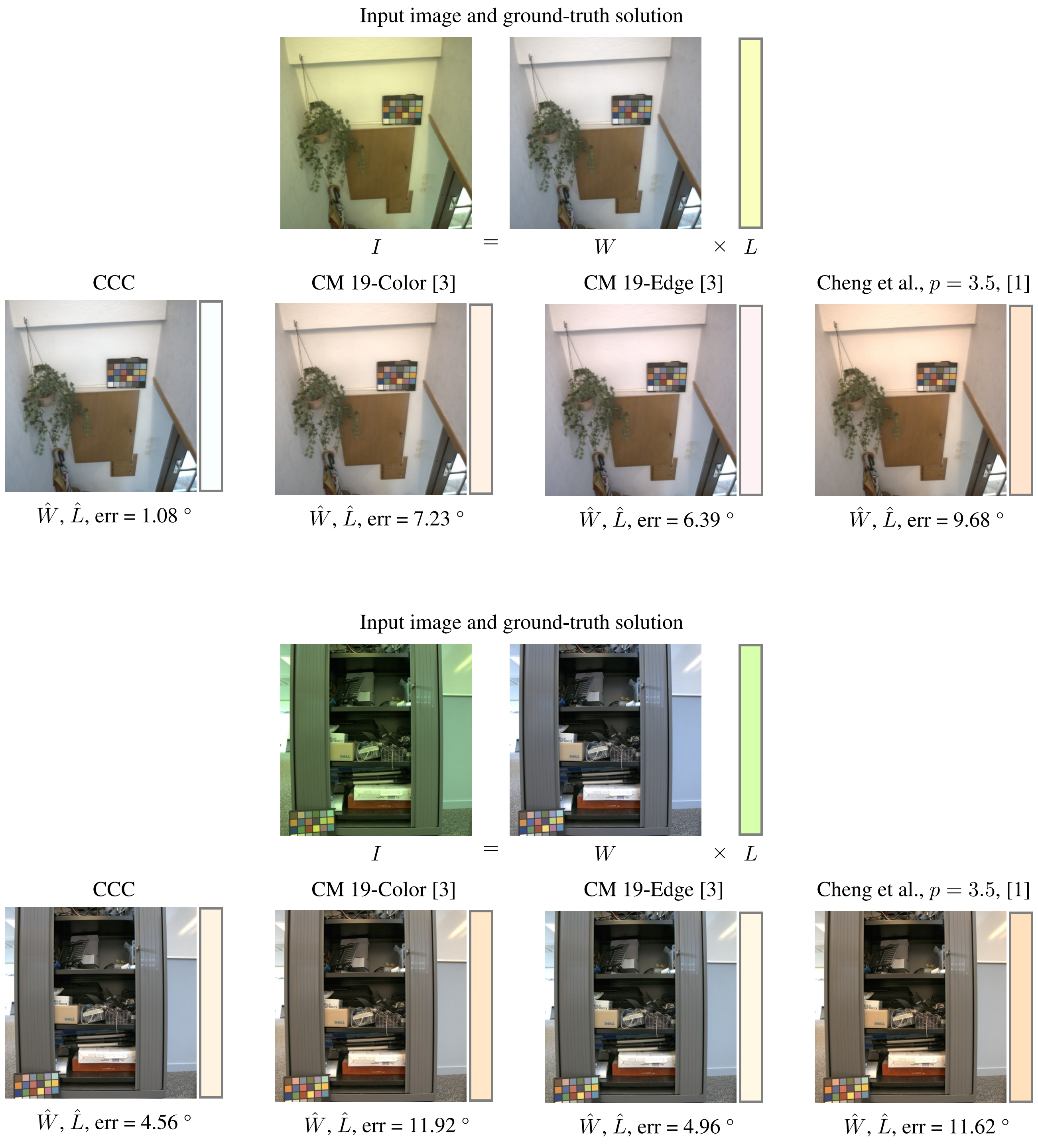
Input image and ground-truth solution
I $=$ W $\times$ L
CCC
CM 19-Color [3]
CM 19-Edge [3]
Cheng et al., $p = 3.5$, [1]
$\hat{W}$, $\hat{L}$, err = 1.08 °
$\hat{W}$, $\hat{L}$, err = 7.23 °
$\hat{W}$, $\hat{L}$, err = 6.39 °
$\hat{W}$, $\hat{L}$, err = 9.68 °
Input image and ground-truth solution
I $=$ W $\times$ L
CCC
CM 19-Color [3]
CM 19-Edge [3]
Cheng et al., $p = 3.5$, [1]
$\hat{W}$, $\hat{L}$, err = 4.56 °
$\hat{W}$, $\hat{L}$, err = 11.92 °
$\hat{W}$, $\hat{L}$, err = 4.96 °
$\hat{W}$, $\hat{L}$, err = 11.62 °

Figure 6: Additional results in the same format as Figure 1.