Consensus Convolutional Sparse Coding Supplemental Material

Biswarup Choudhury *
KAUST
biswarup.choudhury@kaust.edu.sa

Robin Swanson *
KAUST, University of Toronto
robin@cs.toronto.edu

Felix Heide *
Stanford University
fheide@stanford.edu

Gordon Wetzstein
Stanford University
gordon.wetzstein@stanford.edu

Wolfgang Heidrich
KAUST
wolfgang.heidrich@kaust.edu.sa

1. Coefficient Subproblem

As described in the main manuscript in Section 2.1.2, the optimization problem from Equation (4) w.r.t. $\mathbf{z}$ can be written as

$$\underset{\mathbf{z}}{\operatorname{argmin}} \ \frac{1}{2}\|\mathbf{b}' - \mathbf{D}'\mathbf{z}\|_2^2 + \beta\|\mathbf{z}\|_1 \tag{1}$$

where $\mathbf{b}' = [{\mathbf{b}^1}^T \dots {\mathbf{b}^J}^T]^T$, $\mathbf{D}'$ is a block diagonal matrix with $\mathbf{D} = [\mathbf{D}_1 \dots \mathbf{D}_W]$ repeated along its diagonal J times, and $\mathbf{z} = [\mathbf{z}^1 \dots \mathbf{z}^J]^T$ and $\mathbf{z}^j = [{\mathbf{z}_1}^T \dots {\mathbf{z}_W}^T]$. Having defined these operators, we can use the same row partition as in Equation (5) of the main manuscript for $\mathbf{b}'$ and $\mathbf{z}$.

$$\underset{\mathbf{z}_i}{\operatorname{argmin}} \ \frac{1}{2}\sum_{i=1}^{N} \|\mathbf{b}_i - \mathbf{D}\mathbf{z}_i\|_2^2 + \beta\|\mathbf{z}_i\|_1 \tag{2}$$

Analogue to the ADMM method for the filter subproblem, we derive the following Algorithm 3 for solving for the sparse coefficient maps $\mathbf{z}_i$. However, unlike in the filter subproblem, we do not enforce consensus among the coefficient feature maps $\mathbf{z}_i$ since there exists a distinct $\mathbf{z}_i$ for each $\mathbf{b}_i, \forall i = [1 \dots N]$.

In Algorithm 3, each $\mathbf{z}_i$ update takes the form of a Tikhonov-regularized least squares problem, which has the analytical solution

$$\mathbf{z}_i^{k+1} = (\mathbf{D}^\dagger\mathbf{D} + \rho\mathbb{I})^{-1}(\mathbf{D}^\dagger\mathbf{b}_i + \rho(\mathbf{y}_i^k - \lambda_i^k)), \tag{3}$$

where $\cdot^\dagger$ denotes the conjugate transpose. As described in [5, 6, 10] one can find a variable reordering which makes $(\mathbf{D}^\dagger\mathbf{D} + \rho\mathbb{I})$ block-diagonal. Following [10] we directly invert the block-diagonal matrix using Cholesky factorization for the individual blocks in a parallel. The $\mathbf{y}$-update in Algorithm 3 has the form of the shrinkage proximal operator

*Denotes equal contribution

Algorithm 3 ADMM for Feature Map Optimization $\mathbf{z}$

for $i = 1$ to N **do**
  for $k = 1$ to V **do**
    $\mathbf{z}_i^{k+1} = \underset{\mathbf{z}_i}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{b}_i - \mathbf{D}\mathbf{z}_i\|_2^2 + \frac{\rho}{2}\|\mathbf{z}_i - \mathbf{y}_i^k + \lambda_i^k\|_2^2$
    $\mathbf{y}_i^{k+1} = \underset{\mathbf{y}_i}{\operatorname{argmin}} \beta\|\mathbf{y}_i\|_1 + \frac{\rho}{2}\|\mathbf{y}_i - {\mathbf{z}_i}^{k+1} - \lambda_i^k\|_2^2$
    $\lambda_i^{k+1} = \lambda_i^k + \mathbf{z}_i^{k+1} - \mathbf{y}_i^{k+1}$
  end for
end for
$\mathbf{z} = [\mathbf{z}_1^T \dots \mathbf{z}_N^T]^T$

$\mathbf{prox}_{\frac{\beta}{\rho}}({\mathbf{z}_i}^{k+1} + {\lambda_i}^k)$, where

$$\mathbf{prox}_{\theta\|\cdot\|}(\mathbf{v}) = \max\left(1 - \frac{\theta\beta}{|\mathbf{v}|}, 0\right) \odot \mathbf{v} \quad \text{Shrinkage} \tag{4}$$

For a detailed review on proximal operators as a base function of proximal optimization algorithms please refer to [15].

Note that the z-subproblem could be solved using other flavors of parallelized ADMM algorithms. Specifically, we found that solving for $\mathbf{z}_i$ using[12] leads a runtime gain of approximately 20%.

1.1. Splitting Strategies

In our formulation of Equation 4 from the main manuscript the training data $\mathbf{b}$ and coefficient blocks $\mathbf{z}$ were partitioned across the examples (index J). Alternatively, blocks can be chosen *inside* the examples, i.e. blocks for regions in images, when individual example images have dimensions. Another possible partitioning scheme for higher dimensional datasets is to choose blocks across the higher feature dimensions. For example, from a large video dataset, choosing blocks partitioned across the *time* dimension. Both these partitioning schemes allows us to select

Algorithm 4 ADMM for the Consensus Feature Map Optimization $\mathbf{z}$

for $k = 1$ to V **do**
  for $i = 1$ to N **do**
    $\mathbf{z}_i^{k+1} = \underset{\mathbf{z}_i}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{b}_i - \mathbf{D}_i\mathbf{z}_i\|_2^2 + \frac{\rho}{2}\|\mathbf{z}_i - \mathbf{y}^k + \lambda_i^k\|_2^2$
  end for
  $\mathbf{y}^{k+1} = \underset{\mathbf{y}}{\operatorname{argmin}}\ \beta\|\mathbf{y}\|_1 + \frac{N\rho}{2}\|\mathbf{y} - \overline{\mathbf{z}}^{k+1} - \overline{\lambda}^k\|_2^2$
  for $i = 1$ to N **do**
    $\lambda_i^{k+1} = \lambda_i^k + \mathbf{d}_i^{k+1} - \mathbf{y}^{k+1}$
  end for
end for
$\mathbf{z} = \mathbf{y}^{k+1}$

blocks small enough such that the corresponding solver fits into the memory of a single worker. The algorithm for computing the dictionary and their corresponding sparse coefficients maps are Algorithm 1 (main manuscript) and Algorithm 3 respectively.

However, if the partitioning scheme results in overlapping image blocks, then the consensus constraint needs to be enforced across the coefficient maps $\mathbf{z}$ for consistent boundaries. This yields the consensus-based Algorithm 4 (analogoue to Algorithm 1 in main manuscript) for computing the sparse coefficient feature maps $\mathbf{z}$.

1.2. Poisson Noise Penalty

The proximal operator to account for Poisson noise during reconstruction is defined following [9], which is given below for completeness:

$$\mathbf{prox}_\theta(v_i) = \begin{cases} v_i & \mathbf{M}(i) = 1 \\ \frac{v_i - \theta}{2} + \sqrt{\theta\mathbf{b} + \frac{(\theta - v_i)^2}{4}} & else \end{cases}, \tag{5}$$

where $\mathbf{b}$ is the data term we are reconstructing, i indicates a single pixel location in $\mathbf{b}$, and $\mathbf{M}$ represents the logical subsampling mask.

2. Objective Convergence

We have empirically verified the convergence of the proposed algorithm. Figure 1 shows comparisons of existing CSC and CCSC on a small Fruit dataset (10 images), which can entirely fit into available memory. We can observe that the proposed CCSC converges to the same objective as is obtained from the current state-of-the-art CSC technique [10]. Furthermore, we have plotted the convergence for the sequential and the parallel mode of our CCSC algorithm.

Note that the increase in time to converge for our (sequential/parallel) CCSC algorithm is higher because of the additional consensus computations performed on a very small dataset that can easily fit in memory, and therefore can be quickly computed using [10]. Please refer to the main manuscript for runtime comparisons on medium and large datasets which highlight the drastic runtime benefits of the proposed approach.

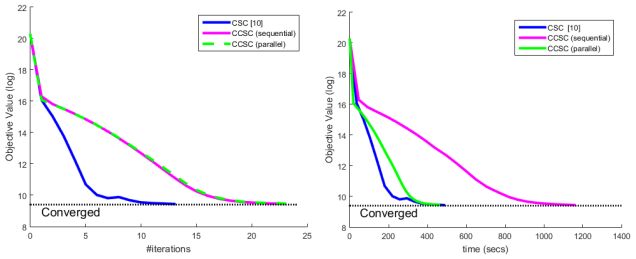


Figure 1: Convergence for sequential block-optimization. When only a single worker is available the subproblems have to be executed sequentially, otherwise they can be parallelized. The plots show convergence for a very small dataset (fruits from [18, 10]). Our consensus optimization method convergences to the same stationary point. Spatial methods such as [18] take several hours (more than 5 for this example) and reach a significantly higher objective.

3. Contrast Normalization

Similar to all existing CSC techniques, we perform contrast normalization on the input data, effective learning from whitened data which is a standard technique in most dictionary learning techniques. However, for the reconstruction of natural images, we require the correct scaling and offset parameters for the image. While 2D CSC methods [5, 10] performed their reconstruction on pre-processed contrast normalized data, Serrano et al. [17] employed complex proximal operators in place of applying contrast normalization directly. We introduce a simple modification to the objective function to implicitly apply contrast normalization during reconstruction.

We introduce the offset term (b') which is our observation data convolved with a local contrast normalization Gaussian kernel [18]. We then modify our objective as,

$$\|\mathbf{b} - \mathbf{M}(\mathbf{D}\mathbf{U}\mathbf{z} + \mathbf{b}')\|_2^2 = \|(\mathbf{b} - \mathbf{M}\mathbf{b}') - \mathbf{M}\mathbf{D}\mathbf{U}\mathbf{z}\|_2^2. \tag{6}$$

Thus allowing us to solve the optimization identically to that described in our main text with only a simple modification of the data term and quadratic proximal operator described in the main text. In addition to its simplicity, this method allows us to reconstruct the original image without affecting channel intensity differences or introducing contrast normalization artifacts.

However, if there is a significant amount of missing data in $\mathbf{b}$ (and therefore $\mathbf{b}'$), inpainting can be computationally

Image	CCSC	[7]	[8]	[2, 3]
Wind Mill	**35.13**	31.98	32.25	23.82
Sea Rock	**28.45**	27.60	27.21	21.20
Parthenon	**31.36**	28.79	28.85	24.75
Rolls Royce	**29.15**	24.47	24.91	21.14
Fence	**30.83**	29.21	29.07	23.67
Car	**34.06**	32.44	32.31	27.19
Kid	**29.41**	27.33	27.26	23.48
Tower	**29.97**	27.96	27.88	23.43
Fish	**31.68**	28.00	28.44	26.39
Food	**36.77**	33.12	33.69	30.14

Table 1: 2D Image Inpainting: CCSC compared to three other state of the art inpainting algorithms. Please note that our algorithm performs much better for all instances as compared to the state of the art inpainting techniques [2, 3, 7, 8].

costly. Therefore, we approximate the missing data in $\mathbf{b}$ before computing $\mathbf{b}'$. Simple interpolation was adequate for all example applications that we explored.

4. Results

4.1. Inpainting

4.1.1 2D Inpainting

In this section, we provide a set of quantitative results (Table 1), comparing our algorithm CCSC with other state of the art inpainting algorithms, followed by qualitative results in Figure 4. We observe that out algorithm performs better than various state of the art inpainting algorithms.

4.1.2 Higher-Dimensional Inpainting

To evaluate CCSC for inpainting in higher dimension, we have reconstructed randomly subsampled multispectral images. The results are shown in Figure 5. To verify that higher-dimensional reconstruction is beneficial, and in particular superior to simply applying 2D CSC on each channel independently, we compared reconstruction results for subsampled multispectral images. In all cases the results of the proposed method were better, often considerably so, than the 2D comparison. This is not all together surprising as we are able to enforce similarity between channels which the 2D methods are not. Quantitative results are listed in Table 2.

4.2. Poisson Deconvolution

We provide 2D Poisson deconvolution qualitative results in Figure 6 for comparison with classical CSC. We also provide quantitative comparison of our CCSC algorithm as compared to state-of-the-art deconvolution algorithms in Table 3.

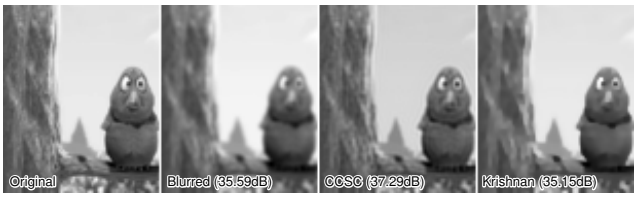
Image	CCSC	[14]	[19]	[16]	[1]
Agama	**28.05**	22.80	24.79	22.71	22.41
Gypful	**29.36**	22.26	24.77	23.60	23.95
Kathmandu	**23.14**	18.10	19.84	19.58	19.34
Laser	**31.59**	25.14	25.77	27.46	26.43
Libelle	**27.56**	18.32	21.55	21.66	18.91
Melinaea	**28.36**	23.19	23.83	23.13	18.81
Mototaxis	**23.41**	19.52	20.31	20.11	19.33
Painted	**22.47**	17.20	17.79	18.61	17.67
Platycercus	**25.93**	18.29	20.10	20.27	20.02
Porsche	**27.11**	18.05	21.14	18.58	17.47

Table 3: Poisson deconvolution: CCSC compared to three other state of the art deconvolution algorithms. Please note that our algorithm performs much better for all instances as compared to the state of the art deconvolution techniques.

4.3. Video Deblurring

We evaluated CCSC when applied to video deblurring. To simulate blurred data we applied a 3×3 snake-like blur kernel to each frame from 10, 100 frame video clips randomly drawn from the "Big Buck Bunny" video[1]. For comparison we have included results from a comparable 2D

[1](c) copyright 2008, Blender Foundation / www.bigbuckbunny.org



Clip	Blurred	CCSC	Krishnan [11]
1	34.94	**40.54**	38.23
2	40.54	**34.41**	32.82
3	38.23	**38.08**	35.59
4	28.65	**39.46**	37.86
5	34.41	**36.85**	36.33
6	32.82	**34.63**	33.91
7	32.53	**36.54**	35.36
8	38.08	**35.83**	33.96
9	35.59	**37.29**	35.15
10	31.49	**35.55**	34.09

Figure 2: Video Deblurring PSNR (dB) Results. Top: Selected deblurred frames. Note that SCSC is able to recover more high frequency information such as the bird feathers and bark speckles. Bottom: Quantitative comparisons on a representative test set of videos.

Dataset	Columbia Cave Testing Set												Harvard Testing Set							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
PSNR Heide [10]	23.17	20.88	17.59	17.06	19.14	20.81	20.66	21.32	21.43	20.28	21.73	22.11	19.40	19.38	19.98	21.51	19.39	19.33	18.88	19.01
PSNR Ours	**25.31**	**27.14**	**29.34**	**37.17**	**23.50**	**31.71**	**33.19**	**32.21**	**30.72**	**22.81**	**25.84**	**25.84**	**39.54**	**40.99**	**49.23**	**31.79**	**36.10**	**41.70**	**28.51**	**46.71**

Table 2: Multispectral Inpainting (2D/3D): Reconstruction quality with our proposed 3D dictionary (bottom row) and the filters learned using [10] (center row). All reconstructions were performed with 50% subsampling of the testing data.

method [11] applied to each frame individually. The results of this comparison and the example output can be found in Figure 2.

4.3.1 Full Video Filters

For completeness we include our learned video dictionary in its entirety in Figure 7.

4.4. Light Field View Synthesis

In Figure 8 we provide our learned light field dictionary in its entirety. We also provide reconstructed novel views for comparison in Figure 9.

4.5. Multispectral Demosaicing

In this section, we include the color filter mosaic used during all of our multispectral demosaicing experiments in Figure 3. Each pixel records information only for a narrow band-pass in wavelength, arranged a filter array of 5×5 pixels.

In addition to the results shown in the main manuscript, we show here the output from both SD and WB algorithms [4], IID [13], and our own in Figure 10. Furthermore,

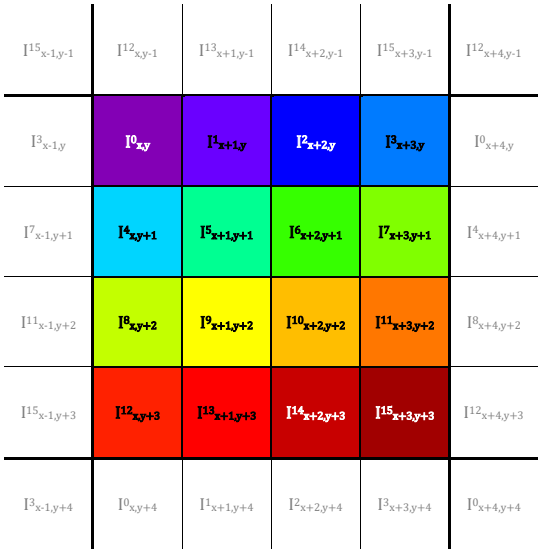


Figure 3: MultiSpectral Filter Array (MSFA) used during demosaicing experiments.

we include synthetic RGB images in Figure 11 which visualize the ability of the demosaicking methods to faithfully capture high frequency detail in the scene.

References

[1] M. S. C. Almeida and M. Figueiredo. Deconvolving images with unknown boundaries using the alternating direction method of multipliers. *IEEE Transactions on Image Processing*, 22(8):3074–3086, 2013. 3

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proc. SIGGRAPH*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000. 3

[3] F. Bornemann and T. März. Fast image inpainting based on coherence transport. *Journal of Mathematical Imaging and Vision*, 28(3):259–278, 2007. 3

[4] J. Brauers and T. Aach. A color filter array based multispectral camera. In *12. Workshop Farbbildverarbeitung*. Ilmenau, 2006. 4

[5] H. Bristow, A. Eriksson, and S. Lucey. Fast convolutional sparse coding. In *Proc. CVPR*, pages 391–398, 2013. 1, 2

[6] H. Bristow and S. Lucey. Optimization methods for convolutional sparse coding. *arXiv:1406.2407*, 2014. 1

[7] M. Elad, J.-L. Starck, P. Querre, and D. Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis. *Applied and Computational Harmonic Analysis*, 19(3):340 – 358, 2005. 3

[8] S. Esedoglu and J. Shen. Digital inpainting based on the mumfordshaheuler image model. *European Journal of Applied Mathematics*, 13(4):353370, 2002. 3

[9] M. A. Figueiredo and J. M. Bioucas-Dias. Deconvolution of poissonian images using variable splitting and augmented lagrangian optimization. In *Statistical Signal Processing, 2009. SSP'09. IEEE/SP 15th Workshop on*, pages 733–736. IEEE, 2009. 2

[10] F. Heide, W. Heidrich, and G. Wetzstein. Fast and flexible convolutional sparse coding. In *Computer Vision and Pattern Recognition, 2015 IEEE Conference on*, pages 5135–5143, 2015. 1, 2, 4

[11] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *Proc. NIPS*, pages 1033–1041, 2009. 3, 4

[12] Z. Lin, R. Liu, and H. Li. Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning. *Machine Learning*, 99(2):287–325, May 2015. 1

[13] S. Mihoubi, O. Losson, B. Mathon, and L. Macaire. Multispectral demosaicing using intensity-based spectral correlation. In *Image Processing Theory, Tools and Applications*

(IPTA), 2015 International Conference on, pages 461–466. IEEE, 2015. 4

[14] J. Pan, Z. Hu, Z. Su, and M. H. Yang. Deblurring text images via l0-regularized intensity and gradient prior. In *Proc. CVPR*, pages 2901–2908, 2014. 3

[15] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231, 2013. 1

[16] D. Perrone and P. Favaro. Total variation blind deconvolution: The devil is in the details. In *Proc. CVPR*, 2014. 3

[17] A. Serrano, F. Heide, D. Gutierrez, G. Wetzstein, and B. Masia. Convolutional sparse coding for high dynamic range imaging. *Computer Graphics Forum*, 35(2), May 2016. 2

[18] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *Proc. CVPR*, pages 2528–2535, 2010. 2

[19] X. Zhu, F. Šroubek, and P. Milanfar. Deconvolving psfs for a better motion deblurring using multiple images. In *Proc. ECCV*, pages 636–647. Springer-Verlag, 2012. 3

Figure 4: 2D Image Inpainting: Additional inpainting results using 2D filters. From left to right: (a) Subsampled image, (b) Ground Truth, (c) CCSC, (d) Conventional CSC. Please zoom into the digital version of this document for pixel-level comparisons or view the attached images themselves.

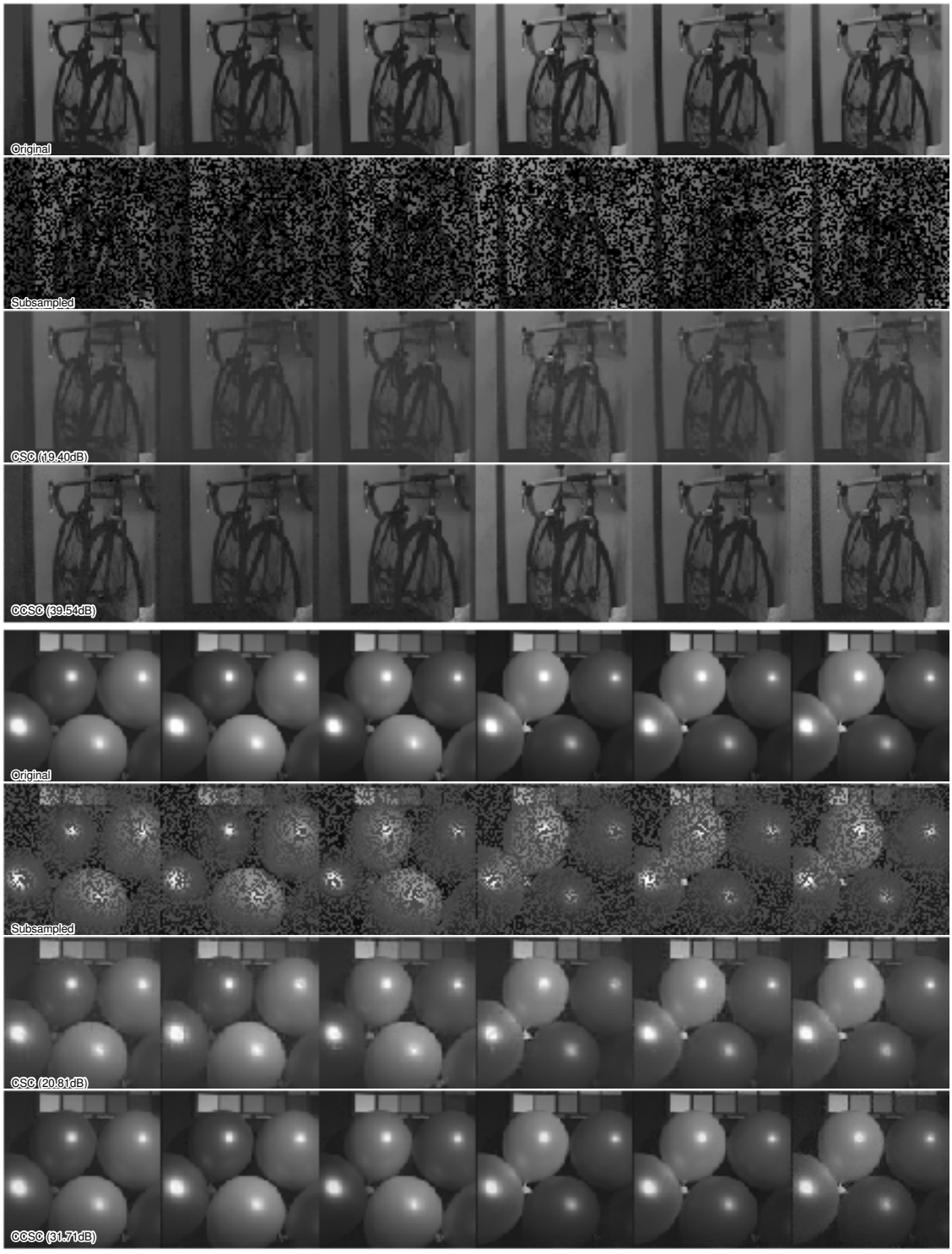


Figure 5: Multispectral Inpainting: Gamma-corrected reconstruction results for the multispectral inpainting problem. For both samples from top row to bottom; (a) Five channels of the original multispectral dataset, (b) Randomly sampled incomplete observations, (c) Reconstruction results with 2D CSC, (d) Reconstruction results with our algorithm. Edges as well as spectral integrity are much better represented by our algorithm which enforces similar correlations across all channels. Please zoom into the digital version of this document for pixel-level comparisons or view the attached images themselves.

Figure 6: Poisson deconvolution: Additional deconvolution results using 2D filters. From left to right: (a) Blurred image, (b) Ground Truth, (c) CCSC, (d) Conventional CSC. Please zoom into the digital version of this document for pixel-level comparisons or view the attached images themselves.

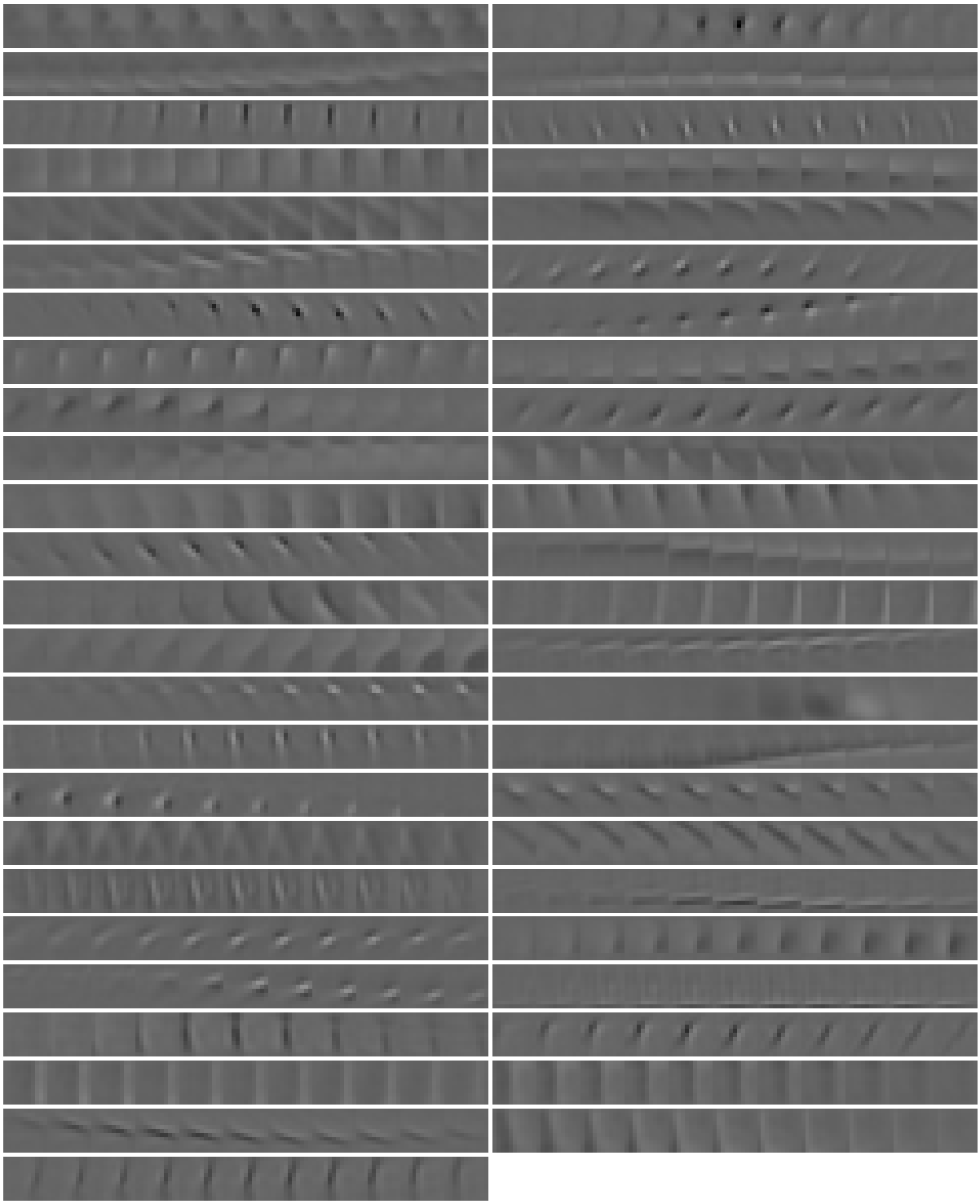


Figure 7: Video Filters: Learned Video Features (3D-Convolutional). Each row shows a single 3D convolutional video kernel whose features slowly change over time from left to right.

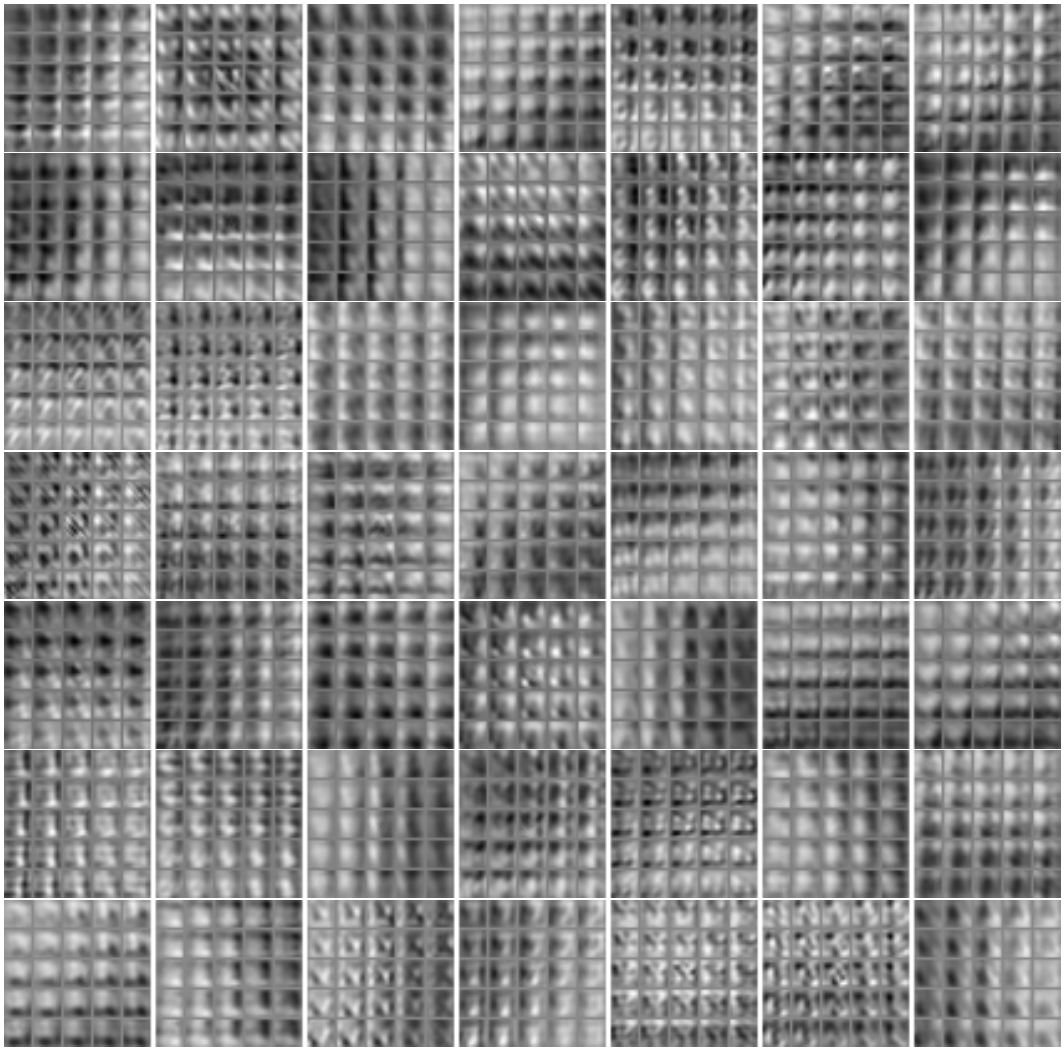


Figure 8: Light Field filters: Learned Light Field Features (2D Convolutional + 2D Non-Convolutional). Each group of 5x5 filters shows all 25 angular features learned.

Figure 9: Light field view synthesis: Five synthesized views from the “Seahorse” data set.

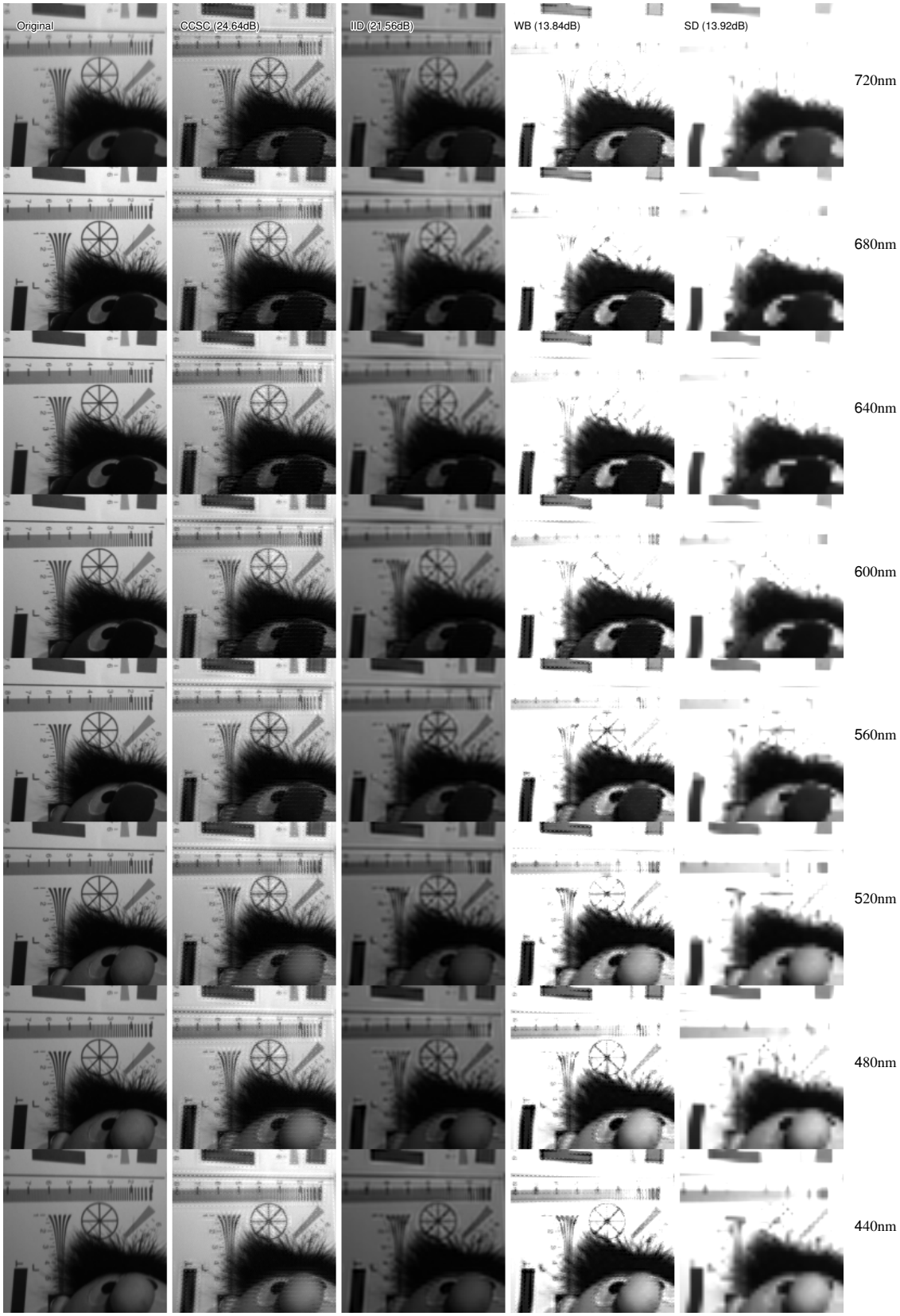


Figure 10: Multispectral demosaicing results from eight wavelengths of the chart dataset.

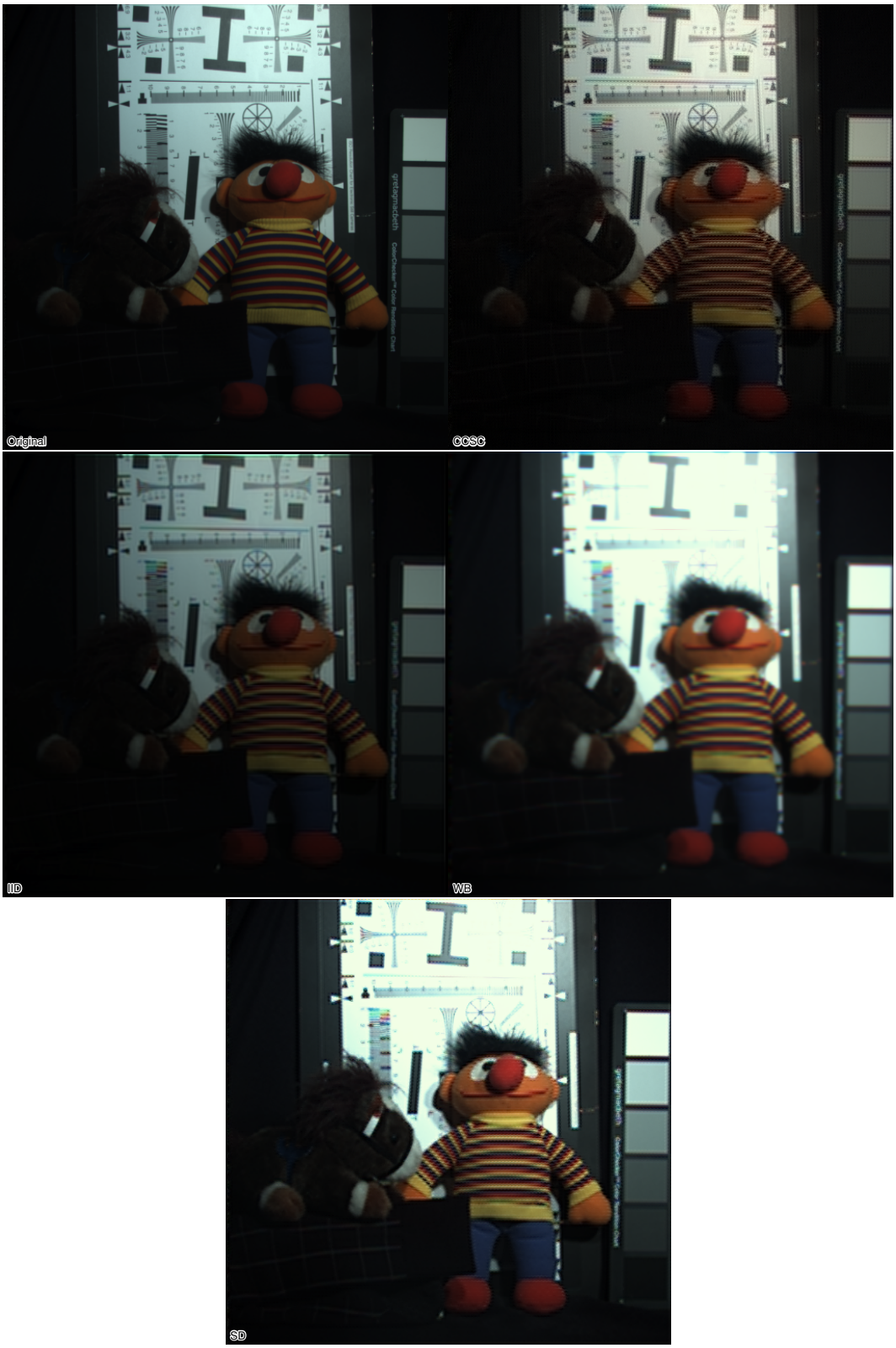


Figure 11: Multispectral demosaicing results visualized as synthetic RGB images.