Consensus Convolutional Sparse Coding

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

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

Felix Heide *
Stanford University
fheide@stanford.edu

Gordon Wetzstein
Stanford University
gordon.wetzstein@stanford.edu

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

Abstract

Convolutional sparse coding (CSC) is a promising direction for unsupervised learning in computer vision. In contrast to recent supervised methods, CSC allows for convolutional image representations to be learned that are equally useful for high-level vision tasks and low-level image reconstruction and can be applied to a wide range of tasks without problem-specific retraining. Due to their extreme memory requirements, however, existing CSC solvers have so far been limited to low-dimensional problems and datasets using a handful of low-resolution example images at a time.

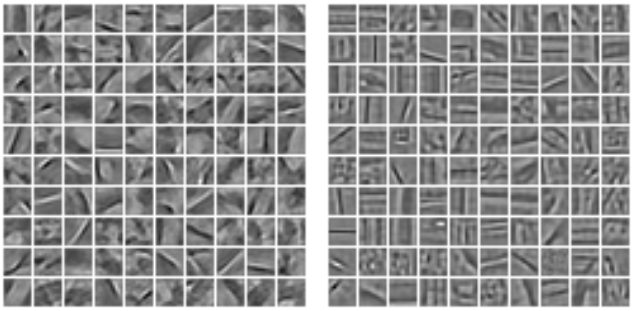
In this paper, we propose a new approach to solving CSC as a consensus optimization problem, which lifts these limitations. By learning CSC features from large-scale image datasets for the first time, we achieve significant quality improvements in a number of imaging tasks. Moreover, the proposed method enables new applications in high-dimensional feature learning that has been intractable using existing CSC methods. This is demonstrated for a variety of reconstruction problems across diverse problem domains, including 3D multispectral demosaicing and 4D light field view synthesis.

1. Introduction

Natural image statistics lie at the core of a wide variety of discriminative and generative computer vision tasks. In particular, convolutional image representations have proven essential for supervised learning using deep neural networks – the de-facto state-of-the-art for many high-level vision tasks [20, 29, 28, 13]. While these models are successful for supervised discriminative problems, the same architectures do not easily transfer to generative tasks.

Generative models have some significant advantages over discriminative models for low level vision and image reconstruction tasks. The most important distinction is that generative approaches learn models of the data that can act as priors for a wide range of reconstruction tasks without retraining, while discriminative methods learn specific reconstruction tasks, and cannot be easily applied to other tasks. As a consequence patch-based sparse coding techniques [7, 23, 1] have been very popular for low-level tasks such as denoising, inpainting, demosaicing, deconvolution and similar problems [11, 34, 30, 24, 21, 2], Unfortunately, patch-based dictionaries are highly redundant because they have to capture all shifted copies of the sparsifying filters.

Introduced as a model for receptive fields in human vision [26], convolution sparse coding (CSC) [14, 17, 32, 33] has been demonstrated to remove much of the overhead of patch-based sparse coding by using a convolution image formation model for a range of different applications [11, 34, 30, 24, 21, 2]. CSC techniques are fast, because many



Traditional CSC Consensus CSC

Figure 1: Large-scale unsupervised feature learning. Left: Convolutional features from [15] can only be learned from a handful of example images since existing CSC methods are limited by memory. Right: CCSC overcomes these limitations, and allows to learn features on ImageNet [9]. These features contain less specialized structures, leading to significant improvements across a variety of vision tasks.

*Denotes equal contribution

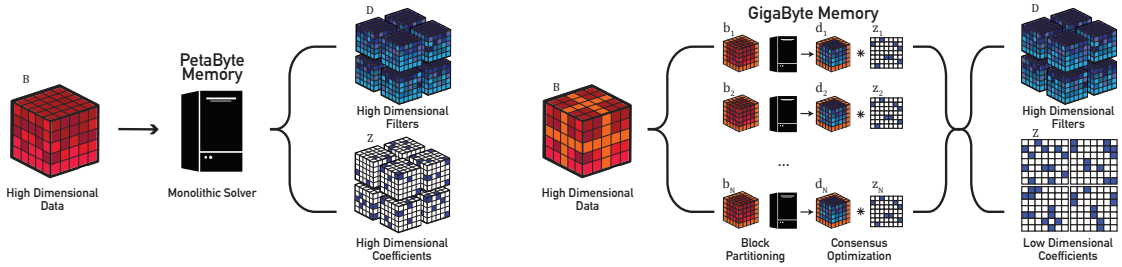


Figure 2: Illustration of traditional CSC (left) and the proposed CCSC (right). CCSC lifts the prohibitive memory limitations of existing algorithms by breaking large, high dimensional datasets into tractable subproblems, each of which can be efficiently solved with a low memory footprint.

implementations efficiently perform convolutions in the frequency domain [5, 6, 15].

While fast, existing CSC approaches are not scalable due to their extreme memory requirements (Fig. 3). For example, existing methods would require terabytes of physical memory for learning light field data from only 100 examples (Sec. 4), and datasets comparable to ImageNet would require petabytes of memory. As a result, it has been intractable to learn convolutional filters from large datasets, and to apply CSC to high-dimensional image reconstruction problems that arise in 3D video, 3D multispectral, or 4D light field image processing.

In this paper, we revisit *unsupervised, generative* learning using CSC, and propose a consensus-based optimization framework that makes CSC tractable on large-scale datasets, and enables high-dimensional feature learning. We call our approach consensus convolutional sparse coding (CCSC). CCSC splits a single large-scale problem into a set of smaller sub-problems that fit into available memory resources. Due to the convex nature of the problem and the enforced consensus between the sub-problems, global convergence is guaranteed. We demonstrate convolutional dictionary learning on datasets that are orders of magnitude larger than what has previously been possible, and show that the resulting sparsifying filters are, in fact, different from those learned from smaller datasets (Fig. 1). Moreover, we show that these new features also lead to significant improvements in a variety of image reconstruction tasks. To validate the proposed method for high-dimensional data, we evaluate CCSC on a number of high-dimensional reconstruction problems that are intractable for existing CSC solvers. In particular, we make the following contributions:

- We derive a consensus optimization method that enables convolutional sparse coding problems of arbitrary size with limited memory to be solved efficiently.
- We extend traditional CSC to allow for non-convolutional data dimensions, greatly reducing memory requirements for high-dimensional datasets.
- We verify the scalability of CCSC by learning from large-scale 2D datasets as well as from several high-dimensional datasets.
- We show that the features learned on large-scale datasets are more general, and lead to better reconstructions than existing methods.
- We evaluate CCSC using several high-dimensional reconstruction problems across diverse problem domains, including 3D multispectral demosaicing, 3D video deblurring, and 4D light field view synthesis.

Finally, the full source code will be made available online for evaluation and improvements in the future.

2. Mathematical Framework

Traditionally, convolutional sparse coding is formulated as the following optimization problem

$$\underset{\mathbf{d},\mathbf{z}}{\operatorname{argmin}} \sum_{j=1}^{J} \frac{1}{2} \|\mathbf{b}^j - \sum_{w=1}^{W} \mathbf{d}_w * \mathbf{z}_w^j\|_2^2 + \beta \sum_{w=1}^{W} \|\mathbf{z}_w^j\|_1 \quad \text{subject to} \quad \|\mathbf{d}_w\|_2^2 \leq 1 \quad \forall\, w \in \{1, \ldots, W\}, \tag{1}$$

where each example image $\mathbf{b}^j$ is represented as the sum of sparse coefficient feature maps $\mathbf{z}_w^j$ convolved with filters $\mathbf{d}_w$ of fixed spatial support. The superscripts indicate the example index $j = 1 \ldots J$, and the subscripts indicate the filter/coefficient map index $w = 1 \ldots W$. The variables $\mathbf{b}^j \in \mathbb{R}^D$ and $\mathbf{z}_w^j \in \mathbb{R}^D$ are vectorized images and feature maps, respectively, $\mathbf{d}_w \in \mathbb{R}^M$ represents the vectorized m-dimensional filters, and $*$ is the m-dimensional convolution operating on the vectorized inputs. The constraint on $\mathbf{d}_w$ ensures the dictionary does not absorb all of the system's energy.

To solve Eq. (1) we first reformulate it as an unconstrained optimization problem, following [15]. Absorbing the constraint in an additional indicator penalty $\mathrm{ind}_C(\cdot)$ for each filter, defined on the convex set of constraints $C = \{x \mid \|Sx\|_2^2 \leq 1\}$, where S is the $\mathbb{R}^{M \times D}$ Fourier submatrix that computes the inverse Fourier transform and projects the result onto the spatial support of each filter, yields

$$\underset{\mathbf{d},\mathbf{z}}{\operatorname{argmin}} \ \frac{1}{2} \sum_j^J (\|\mathbf{b}^j - \mathbf{Z}^j \mathbf{d}\|_2^2 + \beta \|\mathbf{Z}^j\|_1 + \mathrm{ind}_C(\mathbf{d})). \quad (2)$$

Here, $\mathbf{d} = [\mathbf{d}_1^T \dots \mathbf{d}_W^T]^T$, where $\mathbf{d} \in \mathbb{R}^{DW \times 1}$. Similarly, $\mathbf{Z}^j = [\mathbf{Z}_1^j \dots \mathbf{Z}_W^j]$ is a concatenation of Toeplitz matrices, each one expressing the convolution with the respective sparse coefficient map $\mathbf{z}_w^j$ ($\mathbf{Z}^j \in \mathbb{R}^{D \times DW}$). Note that we can express the convolutional term from Eq. (1) in this way because convolution is a commutative operator. Eliminating the sum over the examples (index J) by stacking the vectorized images in $\mathbf{b}' = [{\mathbf{b}_1}^T \dots {\mathbf{b}_J}^T]^T$ and coefficient maps $\mathbf{Z}' = [{\mathbf{Z}^1}^T \dots {\mathbf{Z}^J}^T]^T$ accordingly results in

$$\underset{\mathbf{d},\mathbf{z}}{\operatorname{argmin}} \ \frac{1}{2} \|\mathbf{b}' - \mathbf{Z}'\mathbf{d}\|_2^2 + \beta \|\mathbf{Z}'\|_1 + \mathrm{ind}_C(\mathbf{d}). \quad (3)$$

We jointly solve for both the filters $\mathbf{d}$ and coefficient maps $\mathbf{z}$ in Equation 3 using a coordinate descent approach [15] that alternates between updates to $\mathbf{d}$ and $\mathbf{z}$ while keeping the other fixed (described later in Alg. 2). For this spatial formulation, the filters can be represented in a memory-efficient way, due to their small spatial support. However, the full set of coefficients $\mathbf{z}_w$ must be stored which incurs an enormous memory footprint. Furthermore, convolutions in the spatial domain are computationally expensive.

Recent work [5, 6, 18, 15] has demonstrated that Eq. (3) can be solved efficiently in the frequency domain by applying Parseval's theorem, which states that the energy of a signal is equivalent to that of its Fourier transform up to a constant. In this frequency domain formulation, the previously costly spatial convolutions become efficient Hadamard (component-wise) products. Although computationally efficient, the Fourier formulation still requires frequency representations over the full domain of all frequencies to be held in memory, both for filters and coefficient maps. The size of the coefficient maps grows linearly with the number of filters and images, but exponentially with the dimensionality. For these reasons, classical convolutional sparse coding, and especially its efficient Fourier formulation, do not scale beyond 2D images and small training datasets.

In the following, we derive a consensus optimization method for CSC, allowing to split large-scale and high-dimensional CSC into smaller sub-problems, each of which can be solved with a limited memory budget. Furthermore, the individual sub-problems can be solved efficiently using the Fourier-domain formulation, and in a distributed fashion using parallel workers. Consensus optimization makes CSC tractable for large problems sizes, which we verify by learning from large-scale and high-dimensional datasets.

2.1. Consensus Optimization

To account for large, high-dimensional datasets, we split the problem of learning from the entire dataset $\mathbf{b}'$ into learning from smaller subsets which can be solved individually with modest memory and computational requirements. Specifically, we partition the data vector $\mathbf{b}'$ and their corresponding sparse feature matrix $\mathbf{Z}'$ across all of the examples[1] into N blocks arranged by rows,

$$\mathbf{b}' = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_N \end{bmatrix}, \quad \mathbf{Z}' = \begin{bmatrix} \mathbf{Z}_1 \\ \vdots \\ \mathbf{Z}_N \end{bmatrix}, \quad (4)$$

with $\mathbf{b}_i \in \mathbb{R}^{B_i}$ and $\mathbf{Z}_i \in \mathbb{R}^{B_i \times MW}$, where $\sum_{i=1}^N B_i = JD$. Here, $\mathbf{b}_i$ represents the i^{th} data block along with its respective filters $\mathbf{Z}_i$. In the following we first demonstrate how to solve Eq. (3) using this block splitting with respect to the filters $\mathbf{d}$, and subsequently for the coefficients $\mathbf{z}$.

2.1.1 Filter Subproblem

Using the partition from Eq. (4), we can solve Eq. (3) for $\mathbf{d}$ for a given $\mathbf{Z}'$ as follows

$$\begin{aligned} &\underset{\mathbf{d}}{\operatorname{argmin}} \ \frac{1}{2} \sum_{i=1}^N \|\mathbf{b}_i - \mathbf{Z}_i \mathbf{d}\|_2^2 + \mathrm{ind}_C(\mathbf{d}) \\ \Leftrightarrow \ &\underset{\mathbf{y}}{\operatorname{argmin}} \ \frac{1}{2} \sum_{i=1}^N \|\mathbf{b}_i - \mathbf{Z}_i \mathbf{d}_i\|_2^2 + \mathrm{ind}_C(\mathbf{y}) \\ &\text{subject to} \quad \mathbf{d}_i - \mathbf{y} = 0 \quad \forall i \in \{1, \dots, N\}. \end{aligned} \quad (5)$$

This is a convex problem in the global consensus form [3]. Introducing local variables $\mathbf{d}_i$ allows us to turn the joint objective from the first row of Eq. (5), which cannot be split due to the joint variable $\mathbf{d}$, into separable terms that can be split during the optimization. This also facilitates the handling of the i-th set $(\mathbf{b}_i, \mathbf{Z}_i, \mathbf{d}_i)$ independently by parallel workers. The shared global variable $\mathbf{y} \in \mathbb{R}^{MW}$ introduced as a slack variable enables solving Eq. (5) using the Alternate Direction Method of Multipliers (ADMM) [3, 22], which we derived from the augmented Lagrangian

$$\begin{aligned} \mathcal{L}(\mathbf{d}_1 \dots \mathbf{d}_N, \mathbf{y}, \lambda_1 \dots \lambda_N) = &\sum_{i=1}^N \frac{1}{2} \|\mathbf{b}_i - \mathbf{Z}_i \mathbf{d}_i\|_2^2 \\ &+ \mathrm{ind}_C(\mathbf{y}) + \lambda_i^T (\mathbf{d}_i - \mathbf{y}) + \frac{\rho}{2} \|\mathbf{d}_i - \mathbf{y}\|_2^2, \end{aligned} \quad (6)$$

[1] Please see the supplemental for other splitting strategies.

where λ_i is a set of a Lagrange multipliers for each of the N consensus constraints. ADMM alternately minimizes Eq. (6) with respect to all of its variables, yielding Alg. 1.

Algorithm 1 ADMM for the Filters $\mathbf{d}$

1: **while** Not Converged **do**
2: **for** $i = 1$ to N **do**
3: $\mathbf{d}_i^{k+1} = \underset{\mathbf{d}_i}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{b}_i - \mathbf{Z}_i\mathbf{d}_i\|_2^2 + \frac{\rho}{2}\|\mathbf{d}_i - \mathbf{y}^k + \lambda_i^k\|_2^2$
4: **end for**
5: $\mathbf{y}^{k+1} = \underset{\mathbf{y}}{\operatorname{argmin}}\ \operatorname{ind}_C(\mathbf{y}) + \frac{N\rho}{2}\|\mathbf{y} - \overline{\mathbf{d}}^{k+1} - \overline{\lambda}^k\|_2^2$
6: **for** $i = 1$ to N **do**
7: $\lambda_i^{k+1} = \lambda_i^k + \mathbf{d}_i^{k+1} - \mathbf{y}^{k+1}$
8: **end for**
9: **end while**
10: $\mathbf{d} = \mathbf{y}^{k+1}$

Line 5 uses the average $\overline{\mathbf{d}}^{k+1} = \frac{1}{N}\sum_{i=1}^N \mathbf{d}_i^{k+1}$ and $\overline{\lambda}^k = \frac{1}{N}\sum_{i=1}^N \lambda_i^k$ as a notational shortcut. It becomes clear that the subproblems in the first inner for-loop around Line 3 are now independent of each other. The N subproblems can be solved on a single machine sequentially, or in parallel on up to N workers, each worker i handling only the i-th block of data. After the parallel solve a global synchronization step in Line 5 fuses all individual filter dictionaries, while enforcing the constraint $C = \{x|\ \|Sx\|_2^2 \leq 1\}$. Line 7 updates the Lagrange multipliers for each data-block based on the running error of the fused filters. In the following, we define the individual subproblems of Alg. 1 in detail.

Line 3 is a least-squares problem with the solution

$$\mathbf{d}_i^{k+1} = (\mathbf{Z}_i^\dagger\mathbf{Z}_i + \rho\mathbb{I})^{-1}(\mathbf{Z}_i^\dagger\mathbf{b}_i + \rho(\mathbf{y}^k - \lambda_i^k)), \quad (7)$$

where $\cdot^\dagger$ denotes the conjugate transpose, and $\mathbb{I}$ denotes the identity matrix. As described in [5, 6, 15] one can find a variable reordering which makes $(\mathbf{Z}_i^\dagger\mathbf{Z}_i + \rho\mathbb{I})$ block-diagonal which we directly invert using Cholesky factorization for the individual blocks, in parallel. The update in Line 5 of Alg. 1 is in the form of a proximal operator for which a rich body of literature exists [27]. Specifically, it is $\mathbf{y}^{k+1} = \mathbf{prox}_{\frac{1}{N\rho}}(\overline{\mathbf{d}}^{k+1} + \overline{\lambda}^k)$, with

$$\mathbf{prox}_{\theta\,\operatorname{ind}_C(\cdot)}(\mathbf{v}) = \begin{cases} \frac{\mathbf{Sv}}{\|\mathbf{Sv}\|_2} & : \|\mathbf{Sv}\|_2^2 \geq 1 \\ \mathbf{Sv} & : \text{else} \end{cases} \quad \text{(Projection)} \quad (8)$$

2.1.2 Coefficient Subproblem

The coefficient subproblem can be written as

$$\begin{aligned} &\underset{\mathbf{z}}{\operatorname{argmin}}\ \frac{1}{2}\|\mathbf{b}' - \mathbf{D}'\mathbf{z}\|_2^2 + \beta\|\mathbf{z}\|_1 \\ \Leftrightarrow &\underset{\mathbf{z}_i}{\operatorname{argmin}}\ \frac{1}{2}\sum_{i=1}^N \|\mathbf{b}_i - \mathbf{D}\mathbf{z}_i\|_2^2 + \beta\|\mathbf{z}_i\|_1 \end{aligned}. \quad (9)$$

The sparse coefficient maps $\mathbf{z}$ can be solved analogous to the filters $\mathbf{d}$. This is a result of the convolution from Eq. (1) being commutative, which allows to rewrite $\mathbf{Z}'\mathbf{d} = \mathbf{D}'\mathbf{z}$ in Eq. (3), with $\mathbf{D}'$ is a block diagonal matrix with $\mathbf{D} = blkdiag[\mathbf{D}_1 \dots \mathbf{D}_W]$ repeated along its diagonal J times, and $\mathbf{z} = [\mathbf{z}^1 \dots \mathbf{z}^J]^T$ and $\mathbf{z}^j = [{\mathbf{z}_1^j}^T \dots {\mathbf{z}_W^j}^T]$. Hence, when solving for $\mathbf{z}$, we can follow the recipe from the previous section, using the same block partition. The resulting algorithm can be found in the supplemental material.

2.1.3 Joint Optimization

The previous paragraphs describe optimization methods for solving the joint objective from Eq. (1) for $\mathbf{d}$ and $\mathbf{z}$. We solve for both unknowns jointly by solving the bi-convex optimization problem using coordinate descent, following [5, 15].

Algorithm 2 Large Scale CCSC Learning

1: Initialize parameters $\rho_\mathbf{d} \in \mathbb{R}^+$, $\rho_\mathbf{z} \in \mathbb{R}^+$
2: Initialize variables $\mathbf{d}^0, \mathbf{z}^0, \lambda_\mathbf{d}^0, \lambda_\mathbf{z}^0, \beta$.
3: **repeat**{Outer Iterations}
4: **Filter Update**:
$\mathbf{d}^k, \lambda_\mathbf{d}^k \leftarrow$ Solve with Alg. 1 and $\rho = \rho_\mathbf{d}, \lambda = \lambda_\mathbf{d}^{k-1}$
5: **Coefficient Update**:
$\mathbf{z}^k, \lambda_\mathbf{z}^k \leftarrow$ Detailed in supplemental $\rho = \rho_\mathbf{z}, \lambda = \lambda_\mathbf{z}^{k-1}$
6: **until** No more progress in both directions.

The respective Lagrange multipliers are initialized with those from the previous iteration. ρ is a parameter of the Lagrangian which intuitively is the step size enforcing the Lagrangian step. For any positive ρ, the primal residual $(d_i - y)$ converges to zero, thereby guaranteeing that the algorithm converges to a saddle point. We refer to [3] for a detailed discussion and proof of convergence. Specifically, for our implementation, running the sub-step algorithms for a fixed number of P steps achieved good progress in the coordinate descent step. We terminate the execution when neither sub-step can further decrease the objective.

2.2. Non-Convolutional Dimensions

Above, we have considered all dimensions of the example data $\mathbf{b}$ to be convolutional. However, some image modalities exist only at very low resolution, e.g. the color dimension of an RGB image. In these cases it is common that no convolutional structure can be found. We represent non-convolutional dimensions by introducing an additional replication operator $\operatorname{Rep}(\cdot)$ which repeats the sparse coefficient maps, that do not contain the non-convolutional dimensions, along the missing dimensions. The original convolutional sparse coding problem from Eq. 1 becomes

$$\underset{\mathbf{d},\mathbf{z}}{\operatorname{argmin}} \sum_{j=1}^{J} \frac{1}{2} \|\mathbf{b}^j - \sum_{w=1}^{W} \mathbf{d}_w * \operatorname{Rep}(\mathbf{z}_w^j)\|_2^2 + \beta \sum_{w=1}^{W} \|\mathbf{z}_w^j\|_1$$
$$\text{subject to} \quad \|\mathbf{d}_w\|_2^2 \leq 1 \quad \forall w \in \{1, \ldots, W\}, \tag{10}$$

For example, considering a single dimension with length $\mu = 3$ for RGB image data, $\operatorname{Rep}(\cdot)$ expands the 2D feature-maps to the full three-channel data by replicating the feature map 3 times along the 3rd dimension. The convolution operator is still a 2D convolution, but with full color RGB filters. In Eq. (3), the operator $\operatorname{Rep}(\cdot)$ can be represented by an additional matrix $\mathbf{P} = [\mathbb{I}_1 \ldots \mathbb{I}_\mu]^T$ such that $\mathbf{D}$ and $\mathbf{PZ}$ are then of complimentary dimensions. Redefining the coefficient matrix as $\tilde{\mathbf{Z}} = \mathbf{PZ}$, the described Alg. 1 and 2 generalize to this setting. $\mathbf{P}$ being stacked identity matrices, the efficient inverse from Eq. (7) can be applied.

3. Memory and Complexity Analysis

This section analyzes the memory and runtime of the proposed approach. The consensus optimization from the previous section enables splitting CSC problems of arbitrary size into subproblems that fit into physical memory. Fig. 3 shows the memory consumption of the proposed CCSC approach compared to existing CSC [15], as well as classic patch-based sparse coding [1]. Even on a machine with 128 GB of physcial memory these existing methods become infeasible for learning from medium datasets in 2D, and fail for small data-sets in higher-dimensions. CCSC makes large-scale convolutional sparse coding feasible by efficiently solving smaller subproblems with memory requirements which scale slowly as dataset size and dimensions increase. However, splitting the CSC problem comes at the cost of increased iterations which are necessary to enforce consensus between local variables.

Each subproblem can now be solved sequentially or in parallel, affecting the runtime of the individual iterations. With full parallelization CCSC closely matches classical, non-distributed runtimes, while at the same time allowing CSC to scale. We first present the theoretical computational cost for a single iteration in Figure 4 (top), with P being the number of inner iterations (of the substeps in Alg. 2) and $U \leq N$ being the number of parallel workers. Assuming N blocks of equal size, splitting and distributing drastically reduces the cost of the linear system solves and of the Fourier transforms. In terms of runtime, this smaller per-iteration cost allows more iterations in the proposed consensus optimization, while at the same time enabling scalability in terms of the memory requirements.

In Figure 4 (bottom) we provide empirical evidence of the high computational efficiency of the proposed approach by comparing the best competing CSC technique [15] with

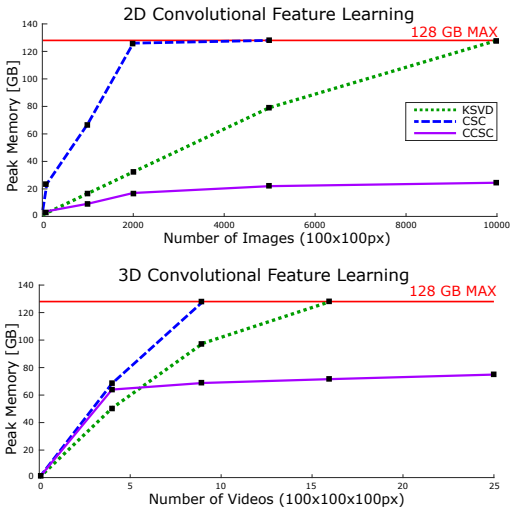


Figure 3: Memory Consumption for large 2D image datasets (top) and video data (bottom). CSC (blue) as well as popular patch-based coding methods (green) become infeasible with increasing size of the dataset (top plot). This effect is even more significant in higher dimensions (bottom plot). Note the very small number of example videos in the bottom plot.

Method	Cost (in flops)
Zeiler et al. [32]	$PJ \cdot (\underbrace{WD}_{\text{Conjugate gradient}} \cdot \underbrace{WDM}_{\text{Spatial convolutions}} + \underbrace{WD}_{\text{Shrinkage}})$
Bristow et al. [5, 6]	$PJ \cdot (\underbrace{W^3D}_{\text{Linear systems}} + \underbrace{WD\log(D)}_{\text{FFTs}} + \underbrace{WD}_{\text{Shrinkage}})$
Heide et al. [15]	$\underbrace{W^3D + (P-1)W^2D}_{\text{Linear systems}} + PJ \cdot (\underbrace{WD\log(D)}_{\text{FFTs}} + \underbrace{WD}_{\text{Shrinkage}})$
CCSC	$\frac{1}{U}\underbrace{(W^3D + (P-1)W^2D)}_{\text{Linear systems}} + \frac{1}{U}PJ \cdot (\underbrace{WD\log(\frac{D}{N})}_{\text{FFTs}} + \underbrace{WD}_{\text{Shrinkage}})$

Dataset Size	CSC [15]	CCSC (U = Number of PCs)		
		U=5	U=10	U=50
100	203.56 sec	35.35 sec	25.69 sec	**25.20 sec**
500	1530.71 sec	259.30 sec	82.69 sec	**28.57 sec**
1000	Out of Memory	387.68 sec	255.38 sec	**35.63 sec**

Figure 4: Complexity and Runtime Analysis. **Top:** Theoretical per-iteration cost of CCSC and other current CSC methods. **Bottom:** Runtime comparisons between the best competing CSC method [15] and CCSC. We demonstrate the runtime gain for a varying number of parallel working threads (U) and increasing dataset size. Note: These values apply only where $\mathbf{Z}$ can be naturally split into equal partitions such that the FFT can be efficiently performed.

CCSC for increasing sizes of 2D dataset with varying number of parallel workers. For example, with a 2D dataset composed of 500 examples (each 100 $\times$ 100 pixels), we observe a speedup of 19$\times$ for 10 workers, and a 54$\times$ for 50 workers over existing CSC methods. Please note that, for datasets of larger size, current CSC techniques are intractable. All algorithms were executed on Intel Xeon 2.7 GHz Dual-core processor with 128GB RAM.

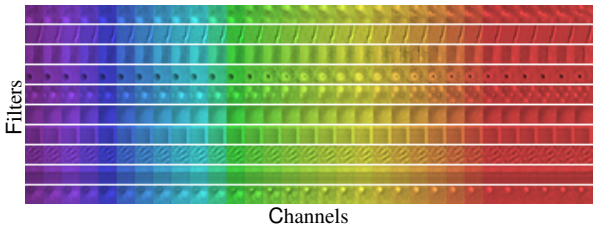


Figure 5: Multispectral (2D convolutional + 1D non-convolutional) dictionary. We show examples of 10 filters learned across all 31 channels on the CAVE dataset. Note the similarity in the kernels across channels which depict the inherent redundancy along multi-channel image data.

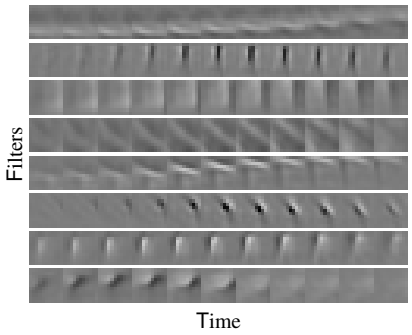


Figure 6: Learned Video Features (3D-Convolutional). Each row shows a single 3D convolutional video kernel whose features slowly change over time from left to right.

4. Learning

Large-scale Feature Learning on ImageNet: To test CCSC on large-scale image data, we use it to learn a dictionary for 5000 images from ImageNet [9] which is at least an order of magnitude more images than previously feasible with CSC methods. The dictionary itself consists of 100 filters of size 11×11, and can be seen in Figure 1. For comparison we have included a similar dictionary trained on a very small fruit dataset. Although superficially similar, the large scale dictionary contains more general features which lead to better reconstruction results (Sec. 5). Our dictionary also contains noise-like filters similar to those learned by discriminative feature learning models [8].

Multi-Spectral Feature Learning: Next, we test CCSC on multispectral data. Each image is now a 3-dimensional entity, with the wavelength as the extra dimension. However, this third dimension is typically much smaller (31 channels in our case) than the two spatial dimensions, and thus we chose to convolve only along the spatial dimensions while the third dimension is non-convolutional in the CCSC dictionary. We therefore force each pixel in the image to share the same coefficients for each element in the dictionary which promotes similarity among all channels without the need for any group sparsity constraints. We found that this method was greatly superior to solving each channel individually with 2D CSC, particularly in the presence of missing data where the proposed method is able to pull information across all channels. For details please refer to the supplementary material.

We trained the dictionaries on a select number of images from the Foster et al. [12] and CAVE [31] hyperspectral datasets, each learning 100, $11\times11\times31$ filters. An example of the CAVE filters can be seen in Figure 5 which show how the proposed framework learns a variety of features that slowly vary from channel to channel.

Video Feature Learning: Unlike multispectral data which contains a fixed number of channels, videos are composed of an arbitrary number of frames which lends itself to a fully convolutional 3D filter. Therefore, we learned a set of 49 3D filters of size $11\times11\times11$ from a varied set of 64 HD video clips. A sample set of these filters can be seen in Figure 6, which demonstrates the variety of CCSC filters as well as their smooth spatial and color transitions across time frames. For reconstruction results please refer to the supplemental material.

Light Field Feature Learning: Although typically captured as a single image, light fields can be represented as a 4D tensor with two spatial dimensions and two angular dimensions. Because the two angular dimensions are small (typically only 5 to 8 angles), we chose to train dictionary filters which were convolutional spatially, but non-convolutional in the angular dimensions. The final dictionary was trained on a set of 64 light fields truncated to 5 angular views in both x and y, and contained 49 filters of size $11\times11\times5\times5$. A sample set of these filters can be found in Figure 7 which clearly demonstrates the angular structure learned by CCSC. Each 5×5 group of filters slowly varies across the angular dimensions while exhibiting general features for reconstruction throughout.

5. Reconstruction

M-Operator: Similar to Heide et al. [15], we employ a binary mask $\mathbf{M}$ as a general linear operator which can be

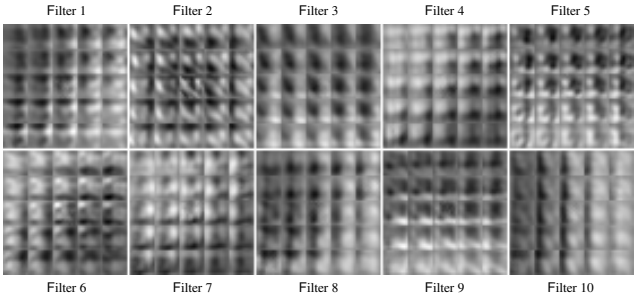


Figure 7: Example of 10 Learned Light Field Features (2D convolutional + 2D non-convolutional). Each group of 5x5 filters shows all 25 angular features learned.

2D Inpainting

Image	CCSC	CSC[15]	NLR [10]
Wind Mill	**35.13**	33.49	27.30
Sea Rock	**28.45**	27.29	23.38
Parthenon	**31.36**	29.79	24.99
Rolls Royce	**29.15**	27.34	22.05
Fence	**30.83**	29.59	22.41
Car	**34.06**	32.57	22.86
Kid	**29.41**	28.05	23.08
Tower	**29.97**	28.30	24.86
Fish	**31.68**	30.26	20.92
Food	**36.77**	35.09	23.78

2D Poisson Deconvolution

Image	CCSC	CSC [15]	Krishnan [19]
Agama	**28.05**	27.87	24.26
Gypful	**29.36**	29.31	24.52
Kathmandu	**23.14**	22.86	20.19
Laser	**31.59**	31.57	28.10
Libelle	**27.56**	27.17	22.86
Melinaea	**28.36**	28.05	24.16
Mototaxis	**23.41**	23.34	20.87
Painted	**22.47**	21.89	18.84
Platycercus	**25.93**	24.94	21.12
Porsche	**27.11**	26.21	19.60

3D Multispectral Demosaicing

Image	CCSC	IID [25]	SD [4]	WB [4]
Balloons	**28.62**	27.38	25.83	25.91
Beads	**23.87**	23.33	14.18	14.57
CD	**30.54**	23.38	18.23	18.39
Chart	**24.64**	21.56	13.92	13.84
Clay	**29.74**	14.25	12.53	12.57
Cloth	**23.50**	20.96	13.91	14.02
Statue	**33.38**	20.83	16.97	17.12
Face	**28.12**	17.50	12.91	13.00
Beer	**23.72**	18.33	9.21	9.23
Food	**28.91**	25.50	17.38	17.61

Figure 8: Quantitative analysis of 2D Image Reconstruction and Multispectral Demosaicing. **Left:** Inpainting results for 50% randomly subsampled observations of images randomly selected from ImageNet [9]. The filters learned using CCSC (shown in Fig. 1) lead to significantly prediction results compared to the ones from [15], as well as recent patch-based methods such as the non-local low-rank method from [10]. **Center:** 2D Poisson Deconvolution. Comparisons of CCSC against the state of the art deconvolution method [19] and the classical CSC method. **Right:** Multispectral Demosaicing results for the CAVE dataset comparing CCSC against the state of the art Iterative Intensity Difference (IID) [25], and the previous standard Spectral Difference (SD) [4] and Weighted Bilinear (WB) [4] interpolation methods. All values reported as PSNR in dB. Please see supplement for comparisons of CCSC with other state of the art techniques.

used for a variety of purposes, such as boundary handling, and masking incomplete data. Note that, typically $\mathbf{M}$ is a diagonal or block diagonal matrix, such that it decouples linear systems of the form $(\mathbf{M}^T\mathbf{M} + \mathbb{I})\mathbf{x} = \mathbf{v}$ into many small independent systems that can be efficiently solved.

Inpainting and Deconvolution: To compare the CCSC large-scale dictionary with conventional CSC, and demonstrate applicability to different noise and image formation models, we evaluated their performance in both inpainting and Poisson noise deconvolution with the Poisson proximal operator described in the supplement. Quantitative results can be found in Figure 8 (left and center), and sample reconstructions can be found in Figure 9 & 10. In all cases the CCSC large-scale features outperformed both classical CSC as well as state of the art alternatives. Please see supplement for additional comparisons of our algorithm with other state of the art techniques.

Multi-Spectral Demosaicing: We compare the proposed method to the state of the art multispectral demosaicing technique [25]. To emulate the demosaicing process we process the raw data to conform to a multispectral filter array (MSFA) pattern with 16 evenly spaced channels corresponding to data from the 400 to 700 nm range. We then reconstruct the data as a sub-sampling problem where the missing data from each channel is masked by the $\mathbf{M}$ operator. We compared the CCSC results with the code provided by [25] on the original CAVE dataset [31] and calculated

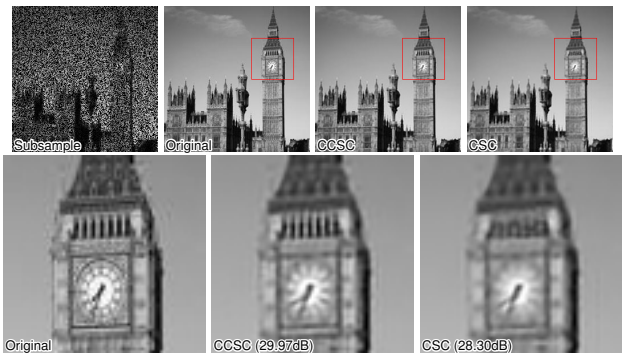


Figure 9: Inpainting results using 2D filters for the "Clock" example. Top row show from left to right: (a) Subsampled image, (b) Ground Truth, (c) CCSC, (d) Conventional CSC. Bottom shows insets from (b-d) respectively. It is evident that conventional filters fail for difficult contrast edges such as the vertical clock features.

Figure 10: Deconvolution results using 2D filters for the Car example. Top row show from left to right: (a) Blurred image, (b) Ground Truth, (c) CCSC, (d) Conventional CSC. Bottom shows insets from (b-d) respectively. In darker regions such as the car text conventional CSC hallucinates features which are not present resulting in poor deconvolution results.

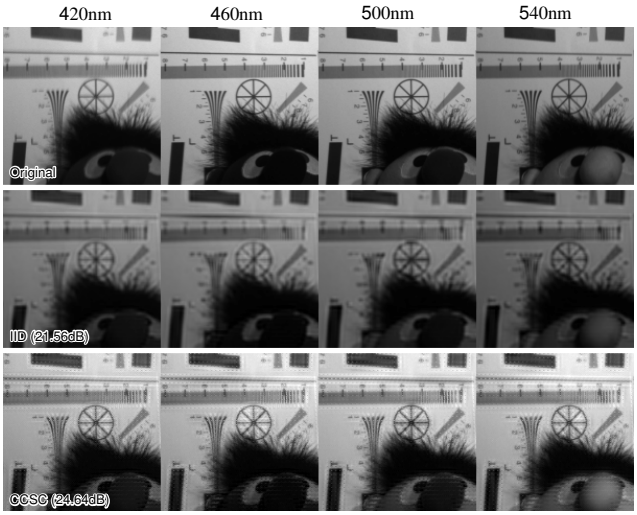
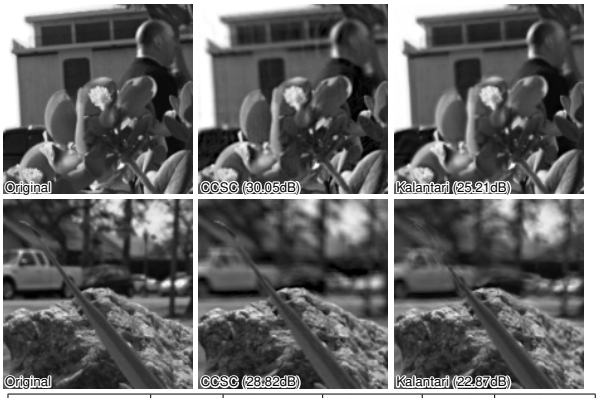


Figure 11: Multispectral demosaicing results from four wavelengths of the chart dataset. Note that while the proposed algorithm does contain demosaicing artifacts it is better able to reconstruct the high frequency details found in the background chart while preserving spectral differences. Find a comparison to additional methods, WB (13.84 dB) and SD (13.92 dB), in the supplement.



Scene	Cars	Flower 1	Flower 2	Rock	Seahorse
CCSC	**27.57**	**30.05**	**29.89**	**28.82**	**31.77**
Kalantari [16]	21.41	25.21	24.13	22.87	25.67

Figure 12: **Top:** Example of synthesized views from the Flower dataset. From left to right, (a) Ground Truth, (b) CCSC, (c) Kalantari [16]. The proposed algorithm produces less noticeable ghosting artifacts due to far away objects and better reconstructs fine detail in nearby objects such as the leaf edges and stalk tip. **Bottom:** Quantitative reconstruction results in PSNR (dB).

the PSNR of the entire reconstructed image. The results in Figure 8 (right) show that CCSC outperforms state of the art techniques, an example of which can be seen in Figure 11.

Light Field View Synthesis: Here we compare CCSC using the learned light field dictionary with state of the art light field view synthesis algorithms. The results can be found in Figure 12 along with sample output. Using the M operator to mask the unknown views we wish to synthesize, we can employ our general reconstruction algorithm to generate the missing data. Using the dictionary described in previous sections with 5×5 angular views and testing data provided by [16], we synthesized the second and fourth angular views in both x and y after removing them from the data. Although this is not the experimental setup used in [16], which may account for some degradation in their performance, it demonstrates the versatility of the proposed approach. One dictionary trained with CCSC can be used to synthesize any number or orientation of light field views.

6. Discussion

Conclusion We have shown that CSC has the potential to be applied in many high and low level computer vision applications. Our distributed CCSC algorithm is both memory efficient and capable of high quality representations of N-Dimensional image data. Furthermore, by reducing and distributing the memory requirements compared to previous CSC methods, our algorithm is capable of handling much larger datasets thereby generating more generalized feature spaces. With our proposed method, we hope to provide a step towards practical and efficient approaches to solving high-dimensional sparse coding problems.

Future Work Although we have shown that CCSC is capable of tackling many computer vision problems, there are many further possible applications. Because our algorithms produce high-dimensional per-pixel coefficients, they could be incorporated into classification, segmentation, or spectral unmixing techniques.

Unlike previous CSC implementations, our distributed framework is amenable to GPU implementation which often have extreme memory constraints. Such an implementation would dramatically increase performance and, for example, bring our multispectral demosaicing algorithm run time in line with other methods.

Acknowledgements: Thanks to Huixuan Tang for discussions, and Katie Black for help with figures. Computer Tower[2] icon by Melvin[3] is licensed under CC-BY 3.0. This work was supported by KAUST baseline funding. Gordon Wetzstein was supported by a Terman Faculty Fellowship, the Intel Compressive Sensing Alliance, the National Science Foundation (IIS 1553333), and the NSF/Intel Partnership on Visual and Experiential Computing (IIS 1539120).

[2]https://thenounproject.com/term/computer-tower/544705/
[3]https://thenounproject.com/nichtcedric/

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. Signal Processing*, 54(11):4311–4322, 2006. 1, 5

[2] N. Akhtar, F. Shafait, and A. Mian. Bayesian sparse representation for hyperspectral image super resolution. In *Proc. IEEE CVPR*, pages 3631–3640, 2015. 1

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011. 3, 4

[4] J. Brauers and T. Aach. A color filter array based multispectral camera. In *12. Workshop Farbbildverarbeitung*. Ilmenau, 2006. 7

[5] H. Bristow, A. Eriksson, and S. Lucey. Fast convolutional sparse coding. In *Proc. CVPR*, pages 391–398, 2013. 2, 3, 4, 5

[6] H. Bristow and S. Lucey. Optimization methods for convolutional sparse coding. *arXiv:1406.2407*, 2014. 2, 3, 4, 5

[7] A. M. Bruckstein, D. L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009. 1

[8] Y. Chen, W. Yu, and T. Pock. On learning optimized reaction diffusion processes for effective image restoration. In *Proc. IEEE CVPR*, pages 5261–5269, 2015. 6

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE CVPR*, pages 248–255, 2009. 1, 6, 7

[10] W. Dong, G. Shi, X. Li, Y. Ma, and F. Huang. Compressive sensing via nonlocal low-rank regularization. *IEEE Trans. Image Processing*, 23(8):3618–3632, 2014. 7

[11] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Processing*, 15(12):3736–3745, 2006. 1

[12] D. H. Foster, K. Amano, S. M. Nascimento, and M. J. Foster. Frequency of metamerism in natural scenes. *JOSA A*, 23(10):2359–2372, 2006. 6

[13] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Proc. IEEE ASSP*, pages 6645–6649. IEEE, 2013. 1

[14] R. B. Grosse, R. Raina, H. Kwong, and A. Y. Ng. Shift-invariance sparse coding for audio classification. In *Proc. UAI*, pages 149–158, 2007. 1

[15] F. Heide, W. Heidrich, and G. Wetzstein. Fast and flexible convolutional sparse coding. In *Proc. IEEE CVPR*, pages 5135–5143, 2015. 1, 2, 3, 4, 5, 6, 7

[16] N. K. Kalantari, T.-C. Wang, and R. Ramamoorthi. Learning-based view synthesis for light field cameras. *arXiv preprint arXiv:1609.02974*, 2016. 8

[17] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierachies for visual recognition. In *Proc. NIPS*, 2010. 1

[18] B. Kong and C. C. Fowlkes. Fast Convolutional Sparse Coding (FCSC). Technical report, UCI, May 2014. 3

[19] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *Proc. NIPS*, pages 1033–1041, 2009. 7

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, pages 1097–1105, 2012. 1

[21] X. Lin, Y. Liu, J. Wu, and Q. Dai. Spatial-spectral encoded compressive hyperspectral imaging. *ACM Trans. Graphics*, 33(6):233, 2014. 1

[22] 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. 3

[23] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proc. ICML*, pages 689–696. ACM, 2009. 1

[24] K. Marwah, G. Wetzstein, Y. Bando, and R. Raskar. Compressive light field photography using overcomplete dictionaries and optimized projections. *ACM Trans. Graph. (SIGGRAPH)*, 32(4):46:1–46:12, 2013. 1

[25] S. Mihoubi, O. Losson, B. Mathon, and L. Macaire. Multispectral demosaicing using intensity-based spectral correlation. In *Proc. IEEE IPTA*, pages 461–466, 2015. 7

[26] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311 – 3325, 1997. 1

[27] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231, 2013. 4

[28] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 1

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[30] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Trans. Image Processing*, 19(11):2861–2873, 2010. 1

[31] F. Yasuma, T. Mitsunaga, D. Iso, and S. K. Nayar. Generalized assorted pixel camera: postcapture control of resolution, dynamic range, and spectrum. *IEEE Trans. Image Processing*, 19(9):2241–2253, 2010. 6, 7

[32] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *Proc. CVPR*, pages 2528–2535, 2010. 1, 5

[33] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Proc. ICCV*, pages 2018–2025, 2011. 1

[34] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *Proc. ICCV*, pages 479–486, 2011. 1