# An MDL Framework for Sparse Coding and Dictionary Learning

Ignacio Ramírez, *Student Member, IEEE*, and Guillermo Sapiro

***Abstract*—The power of sparse signal modeling with learned overcomplete dictionaries has been demonstrated in a variety of applications and fields, from signal processing to statistical inference and machine learning. However, the statistical properties of these models, such as underfitting or overfitting *given* sets of data, are still not well characterized in the literature. As a result, the success of sparse modeling depends on hand-tuning critical parameters for each data and application. This work aims at addressing this by providing a practical and objective characterization of sparse models by means of the minimum description length (MDL) principle—a well-established information-theoretic approach to model selection in statistical inference. The resulting framework derives a family of efficient sparse coding and dictionary learning algorithms which, by virtue of the MDL principle, are completely parameter free. Furthermore, such framework allows to incorporate additional prior information to existing models, such as Markovian dependencies, or to define completely new problem formulations, including in the matrix analysis area, in a natural way. These virtues will be demonstrated with parameter-free algorithms for the classic image denoising and classification problems, and for low-rank matrix recovery in video applications. However, the framework is not limited to this imaging data, and can be applied to a wide range of signal and data types and tasks.**

***Index Terms*—Classification, denoising, dictionary learning, minimum description length principle (MDL) , low-rank matrix completion, sparse coding.**

## I. Introduction

A *SPARSE MODEL* is one in which signals of a given type $\mathbf{y} \in \mathbb{R}^m$ can be represented accurately as sparse linear combinations of the columns (atoms) of a learned dictionary $\mathbf{D} \in \mathbb{R}^{m \times p}$, $\mathbf{y} = \mathbf{D}\mathbf{a} + \mathbf{e}$, where by accurate we mean that $\|\mathbf{e}\| \ll \|\mathbf{y}\|$ (in some norm), and by sparse we mean that the number of nonzero elements in $\mathbf{a}$, denoted by $\|\mathbf{a}\|_0$, is small compared to its dimension $p$. These concepts will be formalized in the next section.

Such models, especially when $\mathbf{D}$ is learned from training samples, are by now a well-established tool in a variety of fields and applications, see [1]–[3] for recent reviews.

When sparsity is a modeling device and not an hypothesis about the nature of the analyzed signals, parameters such as the *desired* sparsity in the solutions, or the size $p$ of the dictionaries to be learned, play a critical role in the effectiveness of sparse models for the data and tasks at hand. However, lacking theoretical guidelines for such parameters, published applications based on learned sparse models often rely on either cross validation or *ad-hoc* methods for determining such critical parameters (an exception for example being the Bayesian approach, e.g., [4]). Clearly, such techniques can be impractical and/or ineffective in many cases. This in turn hinders the further application of such models to new types of data and applications, or their evolution into different, possibly more sophisticated, models.

At the bottom of the aforementioned problem lie fundamental questions such as: *How rich or complex is a sparse model? How does this depend on the required sparsity of the solutions, or the size of the dictionaries? What is the best model for a given data class and a given task?*

The general problem of answering such questions and, in particular, the latter, is known as *model selection*. Popular model selection techniques such as Akaike's Information Criterion (AIC) [5], Bayes Information Criterion (BIC) [6], and the minimum description length principle (MDL) [7]–[9], work by building a cost function which balances a measure of *goodness of fit* with one of *model complexity*, and search for a model $M$ among the set of possible models, or *model class*, $\mathcal{M}$, that minimizes such cost. In this sense, these tools can be regarded as practical implementations of the Occam's razor principle, which states that, given two (equally accurate) descriptions for a given phenomenon, the simpler one is usually the best. Both AIC and BIC propose cost functions of the *penalized likelihood* form. In the case of AIC, the best model for given data $\mathbf{y}$ is the solution to the problem

$$\hat{M} = \arg\min_{M \in \mathcal{M}} -\log \mathrm{P}(\mathbf{y} \mid M) + \mathrm{nfp}(M) \tag{1}$$

where $\mathrm{nfp}(M)$ stands for the *number of free parameters* of the model $M$. This expression is derived as an asymptotic approximation to the Kullback-Leibler divergence between the probability distribution of $\mathbf{y}$ indexed by the model $M$, $\mathrm{P}(\mathbf{y} \mid M)$, and the one based on the maximum-likelihood estimator of $M$ given $\mathbf{y}$, $\hat{M}(\mathbf{y})$ (see [5] for details). In the case of BIC, the best model $\hat{M}$ is chosen via

$$\hat{M} = \arg\min_{M \in \mathcal{M}} -\log \mathrm{P}(\mathbf{y} \mid M) + \frac{1}{2}\mathrm{nfp}(M)\log m \tag{2}$$

where $m$ is the dimension of $\mathbf{y}$. Here the complexity term $\frac{1}{2}\mathrm{nfp}(M)\log m$ is obtained as an asymptotic approximation to the so called *Bayes factor* $\int_{M \in \mathcal{M}} \mathrm{P}(\mathbf{y} \mid M)\mathrm{P}(M)dM$ by means of Laplace's approximation. In MDL, the best model $\hat{M} \in \mathcal{M}$

Manuscript received October 10, 2011; revised January 01, 2012; accepted January 16, 2012. Date of publication February 07, 2012; date of current version May 11, 2012. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Lawrence Carin. This work was supported by the NSF, NGA, ARO, ONR, DARPA, and NSSEFF.

The authors are with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: nacho@fing.edu.uy).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2012.2187203

is the one that can be used to describe $\mathbf{y}$ completely (including the parameters $M$ themselves) with the fewest number of bits

$$\hat{M} = \arg \min_{M \in \mathcal{M}} L(\mathbf{y}, M) \tag{3}$$

where $L(\mathbf{y}, M)$ is a *codelength assignment function* which defines the theoretical codelength required to describe $(\mathbf{y}, M)$ *uniquely*, and which is a key component of any MDL-based framework. The underlying idea of MDL is that *compressibility is a good indirect way of measuring the ability of a model to capture regularity from the data*. Common practice in MDL uses the *Ideal Shannon Codelength Assignment* [10, Ch. 5] to define $L(\mathbf{y}, M)$ in terms of a *probability assignment* $\mathrm{P}(\mathbf{y}, M)$ as $L(\mathbf{y}, M) = -\log \mathrm{P}(\mathbf{y}, M)$ (all logarithms will be assumed on base 2 hereafter). In this way, the problem of choosing $L(\cdot)$ becomes one of choosing a suitable probability model for $(\mathbf{y}, M)$. Note here how MDL considers probability models not as a statement about the true nature of the data, but only as a modeling tool. If we now write $\mathrm{P}(\mathbf{y}, M) = \mathrm{P}(\mathbf{y} \mid M)\mathrm{P}(M)$, we obtain the more familiar penalized likelihood form,

$$\hat{M} = \arg \min_{M \in \mathcal{M}} -\log \mathrm{P}(\mathbf{y} \mid M) - \log \mathrm{P}(M) \tag{4}$$

with $-\log \mathrm{P}(M)$ representing the model complexity, or *model cost*, term. All of the above model selection alternatives have been applied with different degrees of success in a variety of signal processing applications (see, for example, the works on AIC for source detection [11], [12]). This work, for reasons that will become clear in the sequel, focuses on the MDL principle. The use of MDL for sparse signal modeling has been explored for example in the context of wavelet-based denoising (where $M = \mathbf{a} \in \mathbb{R}^m$, $p = m$ and $\mathbf{D} \in \mathbb{R}^{m \times m}$ is *fixed*) of images corrupted by additive white Gaussian noise (AWGN) [13]–[17]. In [13]–[15], the data are described using (4) with $-\log \mathrm{P}(\mathbf{y} \mid \mathbf{a})$ assumed to be *solely due to noise*, and an $L(\mathbf{a})$ term which exploits sparsity

$$\hat{\mathbf{a}} = \arg \min_{\mathbf{a} \in \mathcal{M}} \frac{1}{2\sigma_e^2} \|\mathbf{y} - \mathbf{D}\mathbf{a}\|_2^2 + L(\mathbf{a}). \tag{5}$$

Here the first term corresponds to the ideal codelength, up to a constant, of an i.i.d. Gaussian sequence of zero mean and known variance $\sigma_e^2$. The difference between [13]–[15] lies in the definition of $L(\mathbf{a})$. The line of work [16], [17] follows the modern MDL approach by using sophisticated tools from coding theory, the so called *one-part universal codes*, which encodes $(\mathbf{y}, \mathbf{a})$ jointly, and reduces the arbitrariness in defining $L(\mathbf{a})$. However, such tools can only be applied for certain choices of $\mathrm{P}(\mathbf{y} \mid \mathbf{a})$ and $\mathrm{P}(\mathbf{a})$. In the case of [16], [17], the choice is to use continuous Gaussian models for both. As Gaussian models are *not well suited to the typically observed statistical properties of such data*, the performance of the resulting denoisers for example is very poor compared to the current state-of-the-art.

The work described in this paper includes and extends preliminary results reported in the conference papers [18], [19], using the universal coding models developed in [20]. In particular, new dictionary learning algorithms are developed which include $\ell_1$ atom regularization, forward and backward dictionary size adaptation. All this is done based on the MDL principle as mentioned above. This work extends and/or improves on the aforementioned work [13]–[17] as well as the traditional sparse modeling techniques summarized in [1]–[3], in the following ways:

- MDL-based sparse coding is extended to the case of *nonorthonormal, possibly overcomplete and learned dictionaries* $\mathbf{D}$. As we will see in Section V, this extension, critical to deal with modern, very successful sparse modeling approaches, poses not only new design problems but also significant computational challenges compared to the orthonormal case.
- Efficient codelengths (probability distributions) for the different components to encode (error, coefficients, dictionary) are obtained by *applying universal coding schemes to priors that are suited to the typically observed statistics of such data*.
- As a particular point of the above item, systematic model-fit deviations are naturally taken into account in $\mathrm{P}(\mathbf{y} \mid \mathbf{a})$. The resulting fitting terms fall into the category of robust estimators (see [21]), thus marrying robust statistics with information theory and with sparse modeling (dictionary learning).
- We comply with the basic MDL sanity check, meaning, that *the theoretical codelengths obtained are smaller than a "raw" description of the data*. We do so by including quantization in our models, and treating its effect rigorously.
- Dictionary learning within the MDL framework allows us to *optimize both the number of atoms $p$, as well as their structure*, resulting in a natural and objective form of regularization for $\mathbf{D}$.
- Structure is naturally added to the sparse models in the form of Markovian dependencies between adjacent data samples. We also show an extension of the model to the problem of low-rank matrix completion.

As a result of the above features, we obtain for the first time an MDL-based, parameter-free framework for signal modeling that is able to yield state-of-the-art results. Although the results presented in Section VII for the sake of illustration deal with image and video modeling, the framework here presented is general enough to be applied as-is to a wide range of classes of signals and data, the only underlying assumption being that the input samples are piecewise-continuous signals possibly corrupted by random noise (the basic principles here exploited can be extended beyond this).

At the theoretical level, this brings us a step closer to the fundamental understanding of *learned* sparse models and brings a different perspective, that of MDL, into the sparse modeling world.

The remainder of this paper is organized as follows. Sparse models, and the associated notation, are described in detail in Section II. Section III introduces MDL, and its application to sparse models. In Section IV we present the probability models used to assign codelengths to different parts of the encoded data, while Sections V and VI describe the actual sparse coding and dictionary learning algorithms developed. Experimental

results follow in Section VII, and the paper is concluded in Section VIII.

## II. Background on Sparse Modeling

Assume we are given $n$ $m$-dimensional data samples ordered as columns of a matrix $\mathbf{Y} = [\mathbf{y}_1 \,|\, \mathbf{y}_2| \ldots |\mathbf{y}_n] \in \mathbb{R}^{m\times n}$. Consider a linear model for $\mathbf{Y}, \mathbf{Y} = \mathbf{DA} + \mathbf{E}$, where $\mathbf{D} = [\mathbf{d}_1|\mathbf{d}_2| \ldots |\mathbf{d}_p]$ is an $m \times p$ dictionary consisting of $p$ atoms, $\mathbf{A} = [\mathbf{a}_1|\mathbf{a}_2| \ldots |\mathbf{a}_n] \in \mathbb{R}^{p\times n}$ is a matrix of coefficients where each $j$th column $\mathbf{a}_j$ specifies the linear combination of columns of $\mathbf{D}$ that approximates $\mathbf{y}_j$, and $\mathbf{E} = [\mathbf{e}_1|\mathbf{e}_2| \ldots |\mathbf{e}_n] \in \mathbb{R}^{m\times n}$ is a matrix of approximation errors.

We define the support, or active set, of a vector $\mathbf{a} \in \mathbb{R}^p$ as $\mathrm{supp}(\mathbf{a}) = \{k : \mathbf{a}_k \neq 0\}$. Let $\Gamma = \mathrm{supp}(\mathbf{a})$. We also represent the support of $\mathbf{a}$ as a binary vector $\mathbf{z} \in \{0,1\}^p$ such that $z_i = 1$ for $i \in \Gamma$, and 0 otherwise. We refer to the subvector in $\mathbb{R}^{|\Gamma|}$ of nonzero elements of $\mathbf{a}$ as either $\mathbf{a}_{[\Gamma]}$ or $\mathbf{a}_{[\mathbf{z}]}$. Both conventions are extended to refer to sets of columns of matrices, for example, $\mathbf{D}_{[\Gamma]}$ is the matrix formed by the $|\Gamma|$ columns of $\mathbf{D}$ indexed by $\Gamma$. We will use the pseudo-norm $\|\mathbf{a}\|_0 := |\Gamma| = \sum \mathbf{z}$ to denote the number of nonzero elements of $\mathbf{a}$. We say that the model is *sparse* if we can achieve $\|\mathbf{e}_j\|_2 \ll \|\mathbf{y}_j\|_2$ and $\|\mathbf{a}\|_0 \ll p$ simultaneously for all or most $j = 1, \ldots, n$.

The result of quantizing a real-valued variable $y$ to precision $\delta$ is denoted by $[y]_\delta$. This notation is extended to denote element-wise quantization of vector (e.g., $[\mathbf{e}]$) and matrix operands (e.g., $[\mathbf{E}]$).

### A. Sparse Coding

One possible form of expressing the *sparse coding problem* is given by

$$\hat{\mathbf{a}}_j = \arg\min_{\mathbf{u}\in\mathbb{R}^p} \|\mathbf{y}_j - \mathbf{Du}\|_2 \quad \text{s.t.} \quad \|\mathbf{u}\|_0 \leq \gamma \tag{6}$$

where $\gamma \ll p$ indicates the desired *sparsity level* of the solution. Since problem (6) is nonconvex and NP-hard, approximate solutions are sought. This is done either by using greedy methods such as Matching Pursuit (MP) [22], or by solving a convex approximation to (6), commonly known as the *lasso* [23]

$$\hat{\mathbf{a}}_j = \arg\min_{\mathbf{u}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{y}_j - \mathbf{Du}\|_2 \quad \text{s.t.} \quad \|\mathbf{u}\|_1 \leq \tau. \tag{7}$$

There exists a body of results showing that, under certain conditions on $\gamma$ and $\mathbf{D}$, (6) can be solved exactly via (7) or MP (see, for example, [1] and [24]). In other cases, the objective is not to solve (6), but to guarantee some property of the estimated $\hat{\mathbf{a}}_j$. For example, in the above mentioned case of AWGN denoising in the wavelets domain, the parameter $\tau$ can be chosen so that the resulting estimators are universally optimal with respect to some class of signals [25]. However, if $\mathbf{D}$ is arbitrary, no such choice exists. Also, if $\mathbf{D}$ is orthonormal, the problem (7) admits a closed form solution obtained via the so-called *soft thresholding* [25]. However, again, for general $\mathbf{D}$, no such solution exists, and the search for efficient algorithms has been a hot topic recently, e.g., [26]–[28].

### B. Dictionary Learning

When $\mathbf{D}$ is an optimization variable, we refer to the resulting problem as *dictionary learning*

$$(\hat{\mathbf{A}}, \hat{\mathbf{D}}) = \arg\min_{\mathbf{A},\mathbf{D}} \sum_{j=1}^{n} \frac{1}{2}\|\mathbf{y}_j - \mathbf{Da}_j\|_2^2$$
$$\text{s.t.} \quad \|\mathbf{a}_j\|_r \leq \tau \quad \forall j, \ \|\mathbf{d}_k\|_2 \leq 1 \quad \forall k \tag{8}$$

with $0 \leq r \leq 1$. The constraint $\|\mathbf{d}_k\|_2 \leq 1, \quad k = 1, \ldots, p$, is necessary to avoid an arbitrary decrease of the cost function by setting $\mathbf{D} \leftarrow \alpha\mathbf{D}, \mathbf{A} \leftarrow \frac{1}{\alpha}\mathbf{A}$, for any $\alpha > 1$. The cost function in (8) is nonconvex in $(\mathbf{A}, \mathbf{D})$, so that only local convergence can be guaranteed. This is usually achieved using alternate optimization in $\mathbf{D}$ and $\mathbf{A}$. See, for example, [29], [30], and references therein.

### C. Issues With Traditional Sparse Models: A Motivating Example

Consider the K-SVD-based [29] sparse image restoration framework [31]. This is an $\ell_0$-based dictionary learning framework, which approximates (8) for the case $r = 0$ by alternate minimization. In the case of image denoising, the general procedure can be summarized as follows:

1) An initial, *global* dictionary $\mathbf{D}_0$ is learned using training samples for the class of data to be processed (in this case small patches of natural images). The user must supply a patch width $w$, a dictionary size $p$ and a value for $\tau$.
2) The noisy image is decomposed into overlapping $w \times w$ patches (one patch per pixel of the image), and its noisy patches are used to further adapt $\mathbf{D}$ using the following *denoising* variant of (8)

$$(\hat{\mathbf{D}}, \hat{\mathbf{A}}) = \arg\min_{\mathbf{D},\mathbf{A}} \sum_{j=1}^{n} \|\mathbf{a}_j\|_0,$$
$$\text{s.t.} \quad \frac{1}{2}\|\mathbf{y}_j - \mathbf{Da}_j\|_2^2 \leq C\sigma^2,$$
$$\|\mathbf{d}_k\|_2 = 1, \quad k = 1, \ldots, p. \tag{9}$$

   Here the user must further supply a constant $C$ (in [31], it is 1.32), the noise variance $\sigma^2$, and the number of iterations $J$ of the optimization algorithm, which is usually kept small to avoid overfitting (the algorithm is *not* allowed to converge).
3) The final image is constructed by assembling the patches in $\hat{\mathbf{Y}} = \hat{\mathbf{D}}\hat{\mathbf{A}}$ into the corresponding original positions of the image. The final pixel value at each location is an average of all the patches to which it belongs, plus a small fraction $0 \leq \lambda \leq 1$ of the original noisy pixels ($\lambda = 30/\sigma$ in [31]).

Despite the good results obtained for natural images, several aspects of this method are not satisfactory:

- Several parameters $(w, p, \tau, C, J, \lambda)$ need to be tuned. *There is no interpretation, and therefore no justifiable choice for these parameters, other than maximizing the empirical performance of the algorithm (according to some metric, in this case PSNR) for the data at hand.*
- The effect of such parameters on the result is shadowed by the effects of later stages of the algorithm and their associated parameters (e.g., overlapping patch averaging). *There is no fundamental way to optimize each stage separately.*

The above issues also appear in other applications of traditional sparse models, an example being classification tasks such as those presented in [32] and our previous work [33], where parameters such as $\tau$ and $p$ have to be chosen for each possible class.

As a partial remedy to the first problem, Bayesian sparse models were developed (e.g., [4]) where the unknown parameters are assigned prior distributions which are then learned from the data. For example, whenever possible, the use of the so called "uninformative priors," such as the Jeffreys prior, effectively eliminates model parameters. However, the Bayesian approach still does not provide objective means to compare different models when such models have different functional form (for example, to choose among different possible priors, including uninformative ones, for the parameters), neither does it provide means to determine whether the Bayesian model used is appropriate or not for the class of data at hand. Further, the Bayesian technique implies having to repeatedly solve possibly costly optimization problems, increasing the computational burden of the application (see Section VII-C).[1]

Model selection tools such as AIC [5] or BIC [6] can also be used to select critical parameters such as $\tau$ or $p$. However, with the exception of the computational burden, the limitations of the Bayesian approach also apply to AIC and BIC, since the model complexity terms in their respective cost functions (1) and (2) depends only on the number of free parameters and not on the functional form of the models. In addition, both AIC and BIC are asymptotic approximations to the traditional statistical concept of bias-variance tradeoff (see [34, Ch. 7]), which become valid at very large sample sizes, whereas in typical applications of sparse models such as patch-based image processing, the number of samples is usually very small (e.g., 64 for $8 \times 8$ pixel image patches).

In contrast, MDL allows one to assess the quality of models for representing the given data, by verifying that such models actually yield compressed representations of the data (that is, codelengths that are significantly shorter than a trivial description of the data, such as describing each data sample losslessly). Also, by comparing codelengths, MDL enables the comparison of models which are functionally different, and not only models that belong to the same parametric class, or family of classes. Furthermore, modern MDL provides exact expressions for codelengths for any sample size, and for any data type.

For the above reasons, this work proposes to address the above practical issues, as well as to provide a new angle into dictionary learning, by means of the MDL principle for model selection. The details on how this is done are the subject of the following sections.

## III. Sparse Model Selection and MDL

Given data $\mathbf{Y}$, a maximum support size $\gamma$ and a dictionary size $p$, traditional sparse modeling provides means to estimate the best model $M = (\mathbf{A}, \mathbf{D})$ for $\mathbf{Y}$ within the set $\mathcal{M}(\gamma, p)$ defined as

$$\mathcal{M}(\gamma, p) := \{(\mathbf{A}, \mathbf{D}) : \|\mathbf{a}_j\|_0 \leq \gamma, j = 1, \ldots, n, \mathbf{D} \in \mathbb{R}^{m \times p}\}. \quad (10)$$

[1]The acceleration of Bayesian approaches is currently under investigation and its overall computational time is expected to improve (Prof. Carin, personal communication).

We call such set a *sparse model class* with hyper-parameters $(\gamma, p)$. Such classes are nested in the following way: first, for a fixed dictionary size $p$ we have $\mathcal{M}(\gamma - 1, p) \subset \mathcal{M}(\gamma, p)$. Also, for fixed $\gamma$, if we consider $\mathcal{M}(\gamma, p-1)$ to be a particular case of $\mathcal{M}(\gamma, p)$ where the $p$th atom is all-zeroes and $a_{pj} = 0, \quad \forall j$, then we also have that $\mathcal{M}(\gamma, p-1) \subset \mathcal{M}(\gamma, p)$.

If one wants to choose the best model among all possible classes $\mathcal{M}(\gamma, p)$, the problem becomes one of *model selection*. The general objective of model selection tools is to define an objective criterion for choosing such model. In particular, MDL model selection uses codelength as such criterion. More specifically, this means first computing the best model within each family as

$$(\mathbf{A}(\gamma, p), \mathbf{D}(\gamma, p)) = \arg\min\{L(\mathbf{Y}, \mathbf{A}, \mathbf{D}) : (\mathbf{A}, \mathbf{D}) \in \mathcal{M}(\gamma, p)\}$$

and then choosing $(\hat{\gamma}, \hat{p}) = \arg\min\{L(\mathbf{Y}, \mathbf{A}(\gamma, p), \mathbf{D}(\gamma, p)) : 0 \leq \gamma \leq p,\ p > 0\}$.

When $\mathbf{D}$ is fixed, which is the case of sparse coding, the only model parameter is $\mathbf{A}$, and we have $p + 1$ possible classes, $\mathcal{M}(\gamma) = \{\mathbf{A} : \|\mathbf{a}_j\|_0 \leq \gamma, j = 1, \ldots, n\}$, one for each $0 \leq \gamma \leq p$. If each data sample $\mathbf{y}_j$ from $\mathbf{Y}$ is encoded independently, then, as with traditional sparse coding (the framework can also be extended to *collaborative* models), the model selection problem can be broken into $n$ subproblems, one per sample, by redefining the model class accordingly as $\mathcal{M}(\gamma) = \{\mathbf{a} : \|\mathbf{a}\|_0 \leq \gamma\}$. Clearly, in the latter case, the optimum $\gamma$ can vary from sample to sample.

Compared to the algorithm in Section II-C, we have a *fundamental, intrinsic* measure of the quality of each model, the codelength $L(\mathbf{Y}, \mathbf{A}, \mathbf{D})$, to guide our search through the models, and which is unobscured from the effect of possible later stages of the application. In contrast, there is no obvious intrinsic measure of quality for models learned through (10), making comparisons between models learned for different parameters (patch width $w$, regularization parameter $\tau$, norm $r$, constants $C, \lambda$) possible only in terms of the observed results of the applications where they are embedded. The second advantage of this framework is that it allows to select, in a fundamental fashion, the best model parameters *automatically*, thus resulting in parameter-free algorithms.[2]

Such advantages will be of practical use only if the resulting computational algorithms are not orders of magnitude slower than the traditional ones, and efficient algorithms are a critical component of this framework, see Section V.

### A. A Brief Introduction to MDL

For clarity of the presentation, in this section we will consider $\mathbf{D}$ fixed, and a single data sample $\mathbf{y}$ to be encoded. The MDL principle was pioneered by Rissanen [7] in what is called "early MDL," and later refined by himself [8] and other authors to form what is today known as "modern MDL" (see [35] for an up-to-date extensive reference on the subject). The goal of MDL

[2]For the case of image processing, the patch width $w$ is also a relevant parameter that could be automatically learned with the same MDL-based framework presented here. However, since it is specific to image processing, and due to space constraints and for clarity of the exposition, it will not be considered as part of the model selection problem hereafter.

is to provide an objective criterion to select the model $M$, out of a family of competing models $\mathcal{M}$, that gives the best description of the *given* data $\mathbf{y}$. In this case of sparse coding with fixed dictionary we have $M = \mathbf{a}$.

The main idea of MDL is that, the best model for the data at hand is the one that is able to capture more *regularity* from it. The more regularity a model captures, the more succinct the description of the data will be under that model (by avoiding redundancy in the description). Therefore, MDL will select the best model as the one that produces the shortest (most efficient) description of the data, which in our case is given by $L(\mathbf{y}, \mathbf{a})$.

As aforementioned in Section I, MDL translates the problem of choosing a codelength function $L(\cdot)$ to one of choosing probability models by means of the ideal Shannon codelength assignment $L(\mathbf{y}, \mathbf{a}) = -\log \mathrm{P}(\mathbf{y}, \mathbf{a})$. It is common to extend such ideal codelength to continuous random variables $x$ with probability density function $\mathrm{p}(x)$ as $L(x) = -\log \mathrm{p}(x)$, by assuming that they will be quantized with sufficient precision so that

$$\mathrm{P}([x]_\delta) \approx \mathrm{p}(x)\delta \tag{11}$$

and disregarding the constant term $-\log \delta$ in $L(x)$, as it is inconsequential for model selection. However, in our framework, the optimum quantization levels will often be large enough so that such approximations are no longer valid.

To produce a complete description of the data $\mathbf{y}$, the best model parameters $\hat{M}$ used to encode $\mathbf{y}$ need to be included in the description as well. If the only thing we know is that $\hat{M}$ belongs to a given class $\mathcal{M}$, then the cost of this description will depend on how large and complex $\mathcal{M}$ is. MDL will penalize more those models that come from larger (more complex) classes. This is summarized in one of the fundamental results underlying MDL [8], which establishes that the minimum number of bits required for encoding *any* data vector $\mathbf{y}$ using a model from a class $\mathcal{M}$ has the form $L_{\mathcal{M}}(\mathbf{y}) = \mathcal{L}_{\mathcal{M}}(\mathbf{y}) + \mathcal{C}(\mathcal{M})$, where $\mathcal{L}_{\mathcal{M}}(\mathbf{y})$ is called the *stochastic complexity*, which depends only on the particular instance of $\mathbf{y}$ being encoded, and $\mathcal{C}(\mathcal{M})$ is an unavoidable *parametric complexity* term, which depends *solely* on the structure, geometry, etc., of the model class $\mathcal{M}$.

In the initial version of MDL [7], the parameter $\hat{M}$ was first encoded separately using $L(\hat{M})$ bits, and then $\mathbf{y}$ was described given $\hat{M}$ using $L(\mathbf{y} \mid \hat{M})$ bits, so that the complete description of $\mathbf{y}$ required $L(\mathbf{y} \mid \hat{M}) + L(\hat{M})$ bits. This is called a *two-parts code*. An asymptotic expression of this MDL was developed in [7] which is equivalent to the BIC criterion [6], only in the asymptotic regime. This asymptotic MDL formulation is the one used for example in [11]–[14]. As we will see next, modern MDL departs significantly from this two-parts coding scheme.

### B. Modern MDL and Universal Coding

The main difference between "early" [7] and "modern" [8], [9] MDL is the introduction of *universal codes* as the main building blocks for computing codelengths. In a nutshell, universal coding can be regarded as an extension of the original Shannon theory to the case where the probability model $\mathrm{P}(\cdot)$ of the data to be encoded is not fully specified, but only known to belong to a certain class of candidate probability models $\mathcal{M}$ (recall that classic Shannon theory assumes that $\mathrm{P}(\cdot)$ is perfectly known). For example, $\mathcal{M}$ can be a family of parametric distributions indexed by some parameter $M$. Akin to Shannon theory, for an encoding scheme to be called *universal*, the codelengths it produces need to be optimal, in some sense, with respect to the codelengths produced by all the models in $\mathcal{M}$.

Various universality criteria exist. For example, consider the *codelength redundancy* of a model $\mathrm{Q}(\cdot), \mathcal{R}(\mathbf{y}; Q) = -\log \mathrm{Q}(\mathbf{y}) - [\arg\min_{\mathrm{P} \in \mathcal{M}} - \log \mathrm{P}(\mathbf{y})]$. In words, this is the codelength overhead obtained with $\mathrm{Q}(\cdot)$ for describing an instance $\mathbf{y}$, compared to the best model in $\mathcal{M}$ that could be picked for $\mathbf{y}$, *with hindsight* of $\mathbf{y}$. For example, if $\mathcal{M}$ is a parametric family, such model is given by the maximum likelihood (ML) estimator of $M$. A model $\mathrm{Q}(\cdot)$ is called *minimax universal*, if it minimizes the *worst case redundancy*, $\mathcal{R}(Q) = \arg\max_{\mathbf{y} \in \mathbb{R}^m} \mathcal{R}(\mathbf{y}; Q)$. One of the main techniques in universal coding is *one-part coding*, where the data $\mathbf{y}$ and the best class parameter $\hat{M}$ are encoded jointly. Such codes are used in the line of work of "MDL denoising" due to Rissanen and his collaborators [16], [17]. However, applying one-part codes at this level restricts the probability models to be used.[3] As a consequence, the results obtained with this approach in such works are not competitive with the state-of-the-art. Therefore, in this paper, we maintain a two-parts encoding scheme (or three parts, if $\mathbf{D}$ is to be encoded as well), where we separately describe $\mathbf{a}, \mathbf{D}$, and $\mathbf{y}$ given $(\mathbf{a}, \mathbf{D})$. We will however use universal codes to describe each of these parts as efficiently as possible. Details on this are given in the next section.

## IV. Encoding Scheme

We now define the models and encoding schemes used to describe each of the parts that comprise a sparse model for a data sample $\mathbf{y}$; that is, the dictionary $\mathbf{D}$, the coefficients $\mathbf{a}$, and the approximation error $\mathbf{e} = \mathbf{y} - \mathbf{D}\mathbf{a}$ (which can include both the noise and the model deviation), which can be regarded as the conditional description of $\mathbf{y}$ given the model parameters $(\mathbf{a}, \mathbf{D})$. The result will be a cost function $L(\mathbf{y})$ of the form (note that $\mathbf{y} = \mathbf{D}\mathbf{a} + \mathbf{e}$ can be fully recovered from $(\mathbf{e}, \mathbf{a}, \mathbf{D})$)

$$L(\mathbf{y}, \mathbf{a}, \mathbf{D}) = L(\mathbf{e} | \mathbf{a}, \mathbf{D}) + L(\mathbf{a} \,|\, \mathbf{D}) + L(\mathbf{D}).$$

While computing each of these parts, three main issues need to be dealt with:

1) **Define appropriate probability models**. Here, it is fundamental to incorporate as much prior information as possible, so that no cost is paid in learning (and thereby coding) already known statistical features of the data. Examples of such prior information include sparsity itself, invariance to certain transformations or symmetries, and (Markovian) dependencies between coefficients.
   As mentioned in Section I, although the applications addressed in the experiments of Section VII focus on digital images, the only prior information used in the models here described is that the data to be represented is piecewise-smooth, possibly contaminated with random noise.

[3] In particular, those used in [16], [17] are based on the normalized maximum likelihood (NML) universal model [36], which requires closed-form MLE estimators for its evaluation, something that cannot be obtained for example with a Laplacian prior on $\mathbf{a}$ and nonorthogonal dictionaries.

2) **Deal with unknown parameters**. We will use universal encoding strategies to encode data efficiently in terms of families of probability distributions.
3) **Model the effect of quantization**. All components, $\mathbf{e}, \mathbf{a}, \mathbf{D}$ need to be quantized to some precisions, respectively $\delta_e, \delta_a, \delta_d$, in order to obtain finite, realistic codelengths for describing $\mathbf{y}$ (when the precision variable is obvious from the argument to which it is applied, we drop it to simplify the notation, for example, we will write $[e]_{\delta_e}$ as $[e]$). This quantization introduces several complications, such as optimization over discrete domains, increase of sparsity by rounding to zero, increase of approximation error, and working with discrete probability distributions.

All such issues need to be considered with efficiency of computation in mind. The discussion will focus first on the traditional, single-signal case where each sample $\mathbf{y}$ is encoded separately from the rest. At the end of this section, we will also discuss the extension of this framework to a multi-signal case, which has several algorithmic and modeling advantages over the single-signal case, and which forms the basis for the dictionary learning algorithms described later.

### A. Encoding the Sparse Coefficients

*Probability Model:* Each coefficient in $\mathbf{a}$ is modeled as the product of three (nonindependent) random variables, $A = ZS(V + \delta_a)$, where $Z = 1$ implies $A \neq 0, S = \mathrm{sgn}(A)$, and $V = \max\{|A| - \delta_a, 0\}$ is the absolute value of $A$ corrected for the fact that $V \geq \delta_a$ when $Z = 1$.[4]

We model $Z$ as a Bernoulli variable with $\mathrm{P}(Z = 1) = \rho_a$. Conditioned on $Z = 0, S = V = 0$ with probability 1, so no encoding is needed.[5] Conditioned on $Z = 1$, we assume $\mathrm{P}(S = -1) = \mathrm{P}(S = 1) = 1/2$, and $V$ to be a (discretized) exponential, $\mathrm{Exp}(\theta_a)$. With these choices, $\mathrm{P}(SV \,|\, Z = 1)$ is a (discretized) Laplacian distribution, which is a standard model for transform (e.g., DCT, Wavelet) coefficients. This encoding scheme is depicted in Fig. 1(a), (b). The resulting model is a particular case of the "spike and slab" model used in statistics (see [37] and references therein). A similar factorization of the sparse coefficients is used in the Bayesian framework as well [4].

*Unknown Parameters:* According to the above model, the resulting encoding scheme for the coefficients (sparse code) is a three-parts code: $L(\mathbf{a}) = L(\mathbf{z}) + L(\mathbf{s} \,|\, \mathbf{z}) + L(\mathbf{a} \,|\, \mathbf{s}, \mathbf{z})$. The support $\mathbf{z}$ is described using the *enumerative two-parts code* [38], which first describes its size, $\|\mathbf{a}\|_0$, using $\log p$ bits, and then the particular arrangement of the ones in $\mathbf{z}$ using $\log \binom{p}{\|\mathbf{a}\|_0}$ bits. The total codelength for coding $\mathbf{z}$ is then $L(\mathbf{z}) = \log p + \log \binom{p}{\|\mathbf{a}\|_0}$. This is a universal encoding scheme, and as such is more efficient than those used previously in [13] and [15]. Then, $L(\mathbf{s} \,|\, \mathbf{z}) = \|\mathbf{a}\|_0$ bits are needed to encode $\mathbf{s}_{[\mathbf{z}]}$, the actual signs of the nonzero coefficients. Finally, we need to encode the magnitudes of the $\|\mathbf{a}\|_0$ nonzero coefficients, $\mathbf{a}_{[\mathbf{z}]}$. We do so by considering it first as a sequence of exponentially distributed continuous random variables, to which quantization is applied later. Since the parameter $\theta_a$ of the exponential is unknown,[6] we use a universal model $\mathrm{q}_V(\,\cdot\,)$ for the class of continuous exponential distributions instead. We obtain such universal model $\mathrm{q}_V(V)$ via a convex mixture, one of the standard techniques for this

$$\mathrm{q}_V(V; \kappa_a, \beta_a) = \int_0^{+\infty} \Gamma(\theta; \kappa_a, \beta_a) \frac{\theta}{2} e^{-\theta|V|} d\theta \qquad (12)$$

where the mixing function $\Gamma(\theta; \kappa, \beta) = \Gamma(\kappa)^{-1} \theta^{\kappa-1} \beta^{\kappa} e^{-\beta\theta}$, is the Gamma density function of (noninformative) shape and scale parameters $\kappa$ and $\beta$. With this choice, (12) has a closed form expression, and the degenerate cases $\theta = 0$ and $\theta = \infty$ are given zero weight. The resulting *mixture of exponentials* (MOE) density function $\mathrm{q}_V(V)$, is given by (see [20] for details)

$$\mathrm{q}_V(V; \beta_a, \kappa_a) = \kappa_a \beta_a^{\kappa_a} (V + \beta_a)^{-(\kappa_a+1)}, \quad V \in \mathbb{R}^+.$$

Note that the universality of this mixture model does not depend on the values of the parameters $\kappa_a, \beta_a$, and guided

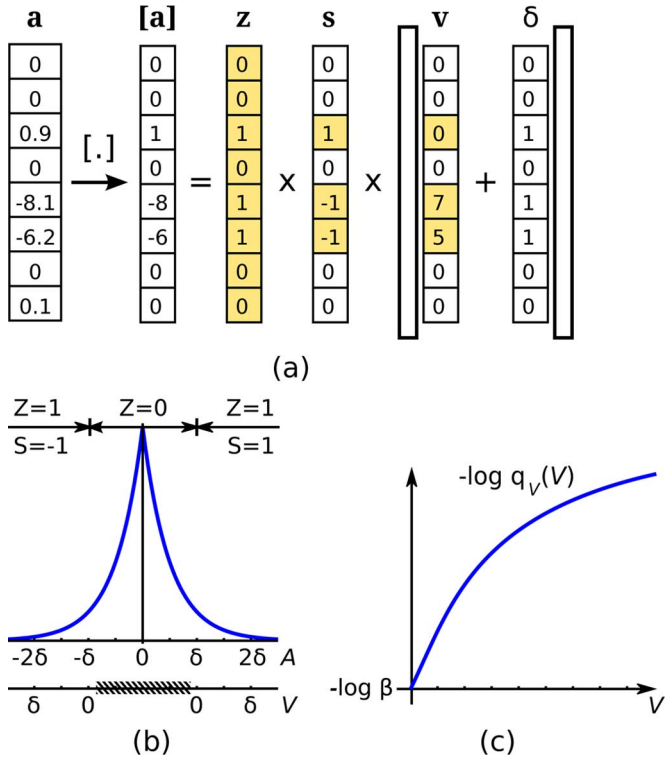


Fig. 1. Encoding of the sparse code. (a) After quantization (here $\delta_a = 1$), each coefficient $a_k$ is decomposed into three variables, $z_k = \mathbf{1}(a_k), s_k = \mathrm{sgn}(a_k)$ and $v_k = \max\{|a_k| - \delta_a, 0\}$. These are respectively modeled by random variables $Z \sim \mathrm{Ber}(\rho_a), S \sim \mathrm{Ber}(1/2), V \sim \mathrm{Exp}(\theta_a)$ (only the shaded numbers are actually encoded). (b) Scheme of the mapping from continuous coefficients (random variable $A$), into $Z, S$ and $V$. (c) Ideal codelength for the MOE model for $V$, $-\log \mathrm{q}_V(V; \kappa, \beta)$. This is a smooth, concave function.

[4]Note that it is necessary to encode $S$ and $V$ separately, instead of considering $SV$ as one random variable, so that the sign of $A$ can be recovered when $|V| = 0$.

[5]We can easily extend the proposed model beyond $S = V = 0$ and consider a distribution for $S, V$ when $Z = 0$. This will naturally appear as part of the coding cost. This extends standard sparse coding to the case where the nonsparse component of the vector are not necessarily zero.

[6]This parameter is related to the sparsity level, and as discussed in Section II-C, is usually assumed known or determined via cross validation. Following [20], here we use tools from universal modeling, which permit to also automatically handle the nonstationarity of this parameter and its expected variability for different nonzero entries of $\mathbf{a}$.

by [20], we set $\kappa_a = 3.0$ and $\beta_a = 50$. The ideal Shannon codelength for this density function distribution is given by $-\log \mathrm{q}_V(V;\kappa_a,\beta_a) = -\log\kappa_a - \kappa_a \log\beta_a + (\kappa_a+1)\log(V+\beta_a)$. This function, shown in Fig. 1(c), is nonconvex, however continuous and differentiable for $V > 0$.

*Quantization:* On one hand, quantizing the coefficients to a finite precision $\delta_a$ increases the approximation/modeling error, from $\mathbf{y} - \mathbf{D}\mathbf{a}$ to $\mathbf{y} - \mathbf{D}[\mathbf{a}]_{\delta_a}$. This additional error, $\mathbf{D}(\mathbf{a} - [\mathbf{a}]_{\delta_a})$, will clearly increase with $\delta_a$. On the other hand, larger $\delta_a$ will reduce the description length of the nonzero values of the coefficients, $[\mathbf{v}]_{\delta_a}$. In practice, for reasonable quantization steps, the error added by such quantization is negligible compared to the approximation error. For example, for describing natural images divided in patches of $8 \times 8$ pixels, our experiments indicate that there is no practical advantage in quantizing the sparse coefficients at a precision smaller than $\delta_a = 16$. Consequently, our current algorithms do not attempt to optimize the codelength on this quantization parameter, and we have kept this value fixed throughout all the experiments of Section VII. Note that this is a data-dependent, but not application-dependent parameter, which would need to be changed only when dealing with other types of data, in particular if the native precision of the input data is significantly different.

### B. Encoding the Error

*Probability Model:* Most sparse coding frameworks, including all the mentioned MDL-based ones, assume the error $\mathbf{e}$ to be solely due to measurement noise, typically of the AWGN type. However, $\mathbf{e}$ actually contains a significant component which is due to a systematic deviation of the model from the clean data. Following this, we model the elements of $\mathbf{e}$ as samples of an i.i.d. random variable $E$ which is the linear combination of two independent variables, $E = \hat{E} + N$. Here $N \sim \mathcal{N}(0,\sigma_e^2)$ represents random measurement noise in $\mathbf{y}$. We assume the noise variance $\sigma_e^2$ known, as it can be easily and reliably estimated from the input data (for example, taking the minimum empirical variance over a sufficient number of subsamples). The distribution of the second variable, $\hat{E} \sim \mathrm{Lap}(0,\theta_e)$ is a Laplacian of unknown parameter $\theta_e$, which represents the error component due to the model itself. The resulting continuous distribution $\mathrm{p}_E(E)$, which we call "LG," is the convolution of the distributions of both components (see [39] for details on the derivation),

$$\begin{aligned}\mathrm{p}_E\left(E;\sigma_e^2,\theta_e\right) &= \int_{\zeta=-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma_e^2}} e^{-\frac{\zeta^1}{2\sigma_e^2}} \frac{1}{2\theta_e} e^{-\frac{|E-\zeta|}{\theta_e}} d\zeta \\ &= \frac{1}{4\theta_e} e^{\frac{\sigma_e^2}{2\theta_e^2}} \left[ e^{E/\theta_e} \mathrm{erfc}\left(\frac{E+\sigma_e^2/\theta_e}{\sqrt{2}\sigma_e}\right) \right. \\ &\quad \left. + e^{-E/\theta_e} \mathrm{erfc}\left(\frac{-E+\sigma_e^2/\theta_e}{\sqrt{2}\sigma_e}\right)\right] \end{aligned} \quad (13)$$

where $\mathrm{erfc}(u) = \frac{2}{\sqrt{\pi}} \int_u^{+\infty} e^{-t^2} dt$ is the *complimentary Gauss error function*. The ideal codelength, $-\log \mathrm{p}_E(E)$, is shown in Fig. 2(a) for various parameter values. This function is convex and differentiable on $\mathbb{R}$, which is nice for optimization purposes. Fig. 2(b) shows its derivative, or so called "influence function" in robust statistics. It can be verified that $-\log \mathrm{p}_E(E)$ behaves like a Laplacian with parameter $\theta_e$ for large values of $E$. Further, since its derivative is bounded, the influence of outliers is diminished. In fact, $-\log \mathrm{p}_E(E)$ is easily verified to be a $\psi$-type M-estimator, a family of functions used in robust statistics (see [21]). Thus, using this model, we obtain an information-theoretic robust estimator, which is consistent with the motivations leading to its use in our framework, and which has a significant practical impact in the experimental results.

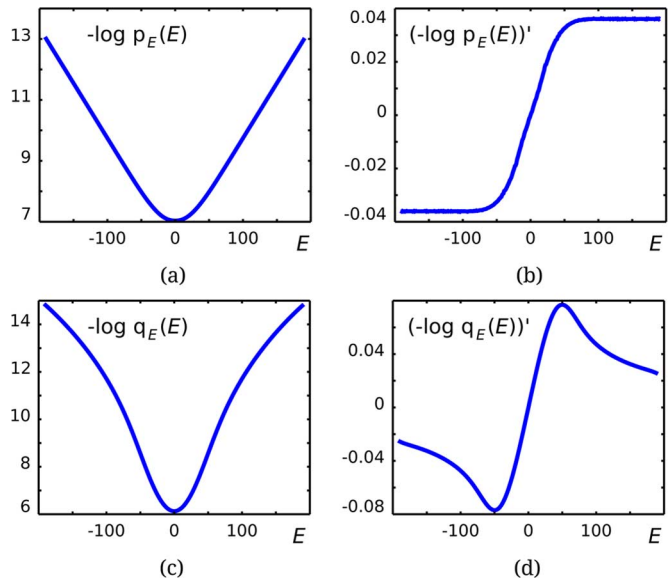


Fig. 2. Residual probability model. (a) Ideal codelength function of the "LG" distribution, $-\log \mathrm{p}_E(E)$. (b) LG influence function, that is, $(-\log \mathrm{p}_E(y))'$. (c) universal mixture for the LG model (MOEG). (d) MOEG influence function.

*Unknown Parameters:* Since $\theta_e$ is unknown, encoding $\mathbf{e}$ efficiently calls for the use of universal codes. In this case, again, we employ a mixture model. Since the parameter $\theta_e$ comes from the underlying Laplacian component, we again use a Gamma for the mixing function

$$\mathrm{q}_E(E;\sigma_e^2,\kappa_e,\beta_e) = \int_0^{+\infty} \Gamma(\theta;\kappa_e,\beta_e)\mathrm{p}_E(E;\sigma_e^2,\theta)d\theta. \quad (14)$$

We call this model MOEG. As with the MOE model, the universality of this model is guaranteed by the theory for the choice of its underlying mixing function, for any (noninformative) $\kappa_e$ and $\beta_e$. In this case, we use $\kappa_e = 3.0$ and $\beta_e = \delta_e$. Also, we know from the discussion above that $\sigma_e^2$ can be easily and reliably estimated from the data. Thus, we can say that the model for $E$ is parameter-free in this case as well. Fig. 2(c) shows the numerical evaluation of the ideal Shannon codelength $-\log \mathrm{q}_E(E;\sigma_e^2,\kappa_e,\beta_e)$, which is nonconvex. However, it is twice differentiable everywhere, again a desirable property for optimization purposes (more on this in Sections V and VI). As with the LG distribution, $-\log \mathrm{q}_E(E)$ is an $\psi$-type M-estimator, in this case, a *redescending* M-estimator, since its derivative [Fig. 2(d)] vanishes to 0 at $\infty$. As such, $-\log \mathrm{q}_E(E)$, derived from the universal model corresponding to $\mathrm{p}_E(E)$, can reject outliers even more aggressively than $-\log \mathrm{p}_E(E)$, again marrying robust statistics with information theory in a natural way.

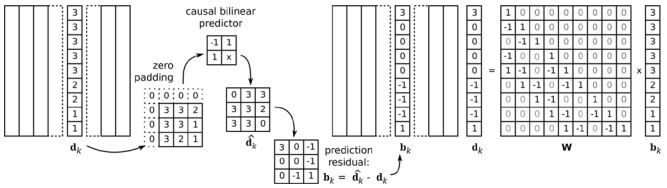


Fig. 3. Prediction scheme used for learning natural image patches dictionaries (in this example, $3 \times 3$ patches, and $m = 9$). An atom $\mathbf{d}_k$ is arranged as a $3 \times 3$ patch, and a causal bilinear predictor (shown as a $2 \times 2$ template) with zero-padding (pixels outside of the patch are assumed 0) is applied to it, producing a predicted atom $\hat{\mathbf{d}}_k$ and a residual $\mathbf{b}_k = \mathbf{d}_k - \hat{\mathbf{d}}_k$. The previous operation can be written as $\mathbf{b}_k = \mathbf{W}\mathbf{d}_k$, with $\mathbf{W} \in \mathbb{R}^{9\times 9}$ the linear mapping from atom to prediction residuals corresponding to this example.

*Quantization:* To losslessly encode finite-precision input data such as digital images, the quantization step of the error coefficients needs not be more than that of the data itself $\delta_y$, and we simply quantize the error coefficients uniformly with step $\delta_e = \delta_y$. For example, for 8-bit digital images, we set $\delta_e = \delta_y = 1$.

### C. Model for the Dictionary

*Probability Model:* Dictionary learning practice shows that learned atoms, unsurprisingly, present features that are similar to those of the original data. For example, the piecewise smoothness of small image patches is to be expected in the atoms of learned dictionaries for such data. This prior information, often neglected in dictionary learning algorithms, needs to be taken into account for encoding such atoms efficiently.

We embody such information in the form of *predictability*. This is, we will encode an atom $\mathbf{d} \in \mathbb{R}^m$ as a sequence of causal prediction residuals, $\mathbf{b} \in \mathbb{R}^m, b_{i+1} = d_{i+1} - \tilde{d}_{i+1}(d_1, d_2, \ldots, d_i)$, $1 \leq i < m$, a function of the previously encoded elements in $\mathbf{d}$. In particular, if we restrict $\tilde{d}_{i+1}$ to be a linear function, the residual vector can be written as $\mathbf{b} = \mathbf{W}\mathbf{d}$, where $\mathbf{W} \in \mathbb{R}^{m\times m}$ is lower triangular due to the causality constraint (this aspect has important efficiency consequences in the algorithms to be developed in Section VI). This is depicted in Fig. 3, along with the specific prediction scheme that we adopted for the image processing examples in Section VII. In this case we consider an atom $\mathbf{d}$ to be an $\sqrt{m} \times \sqrt{m}$ image patch, and use a causal bilinear predictor where the prediction of each pixel in the dictionary atom is given by $\mathrm{north\_pixel} + \mathrm{west\_pixel} - \mathrm{northwest\_pixel}$.

As a general model for linear prediction residuals, we assume $\mathbf{b}$ to be a sequence of i.i.d. Laplacian samples of parameter $\theta_d$. In principle, $\theta_d$ is also unknown. However, describing $\mathbf{D}$ is only meaningful for dictionary learning purposes, and, in that case, $\mathbf{D}$ is updated iteratively, so that when computing an iterate $\mathbf{D}^{(t)}$ of $\mathbf{D}$, we can use $\mathbf{D}^{(t-1)}$ to estimate and fix $\theta_d$ via ML (more on this $\theta_d$ later in Section VI). Thus, we consider $\theta_d$ to be known.

*Quantization:* When $\mathbf{A}$ is fixed during a dictionary learning iteration (which consists of an alternate descent between $\mathbf{D}$ and $\mathbf{A}$), we can view $(\mathbf{A}, \mathbf{Y})$ as $n$ input-output training pairs, and $\mathbf{D}$ as the ML estimator of the linear coefficients describing such mapping via $\mathbf{Y} = \mathbf{D}\mathbf{A} + \mathbf{E}$. Based on this, we use the quantization step $\delta_d = 1/\sqrt{n}$, which is an optimal step for encoding the ML parameter in two-part codes, as described in Theorem 1.

*Computation:* Computing $L(\mathbf{D})$ is only relevant for learning purposes. In general, since $\|\mathbf{d}_k\|_2 \leq 1$, and $\|\mathbf{d}_k\|_2 \leq \sqrt{m}\|\mathbf{d}_k\|_1$, we have that $\hat{\theta}_d = (pm)^{-1}\sum_k \|\mathbf{d}_k\|_1 \leq (p\sqrt{m})^{-1} \ll \delta_d = \sqrt{n}$, and the error of using the approximation (11) is not significant

$$\begin{aligned} L(\mathbf{D}) &= \sum_{k=1}^{p} L(\mathbf{d}_k) \\ &\approx \sum_{k=1}^{p} \{-\log \mathrm{p}(\mathbf{W}\mathbf{d}_k; \theta_d) - m \log \delta_d\} \\ &= \theta_d \sum_{k=1}^{p} \|\mathbf{W}\mathbf{d}_k\|_1 + \frac{mp}{2}\log n + c \end{aligned} \tag{15}$$

where $\mathrm{p}(\mathbf{W}\mathbf{d}_k)$ is the i.i.d. Laplacian distribution over the $k$th atom prediction residual vector $\mathbf{W}\mathbf{d}_k$, and $c$ is a fixed constant. For $p$ fixed (we will later see how to learn the dictionary size $p$ as well), the above expression is simply an $\ell_1$ penalty on the atom prediction residual coefficients. As we will see in Section VI, this allows us to use efficient convex optimization tools to update the atoms.

### D. Extension to Sequential (Collaborative) Coding

One natural assumption that we can often make on the set of data samples $\mathbf{Y}$ is that, besides all being sparsely representable in terms of the learned dictionary $\mathbf{D}$, they share other statistical properties. For example, we can assume that the underlying unknown model parameters, $\theta_e, \rho_a, \theta_a, \theta_d$, are the same for all columns of the sparse data decomposition $(\mathbf{E}, \mathbf{A})$.

Under such assumption, if we encode each column of $\mathbf{Y}$ sequentially, we can learn statistical information from the ones already encoded and apply it to estimate the unknown parameters of the distributions used for encoding the following ones. The general idea is depicted in Fig. 4(a). Concretely, suppose we have already encoded $j-1$ samples. We can then use $[\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{(j-1)}]$ to estimate $\theta_e$, and $[\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{(j-1)}]$ to estimate $\theta_a$ and $\rho_a$, and "plug-in" these parameters to encode the $j$th sample. This justifies the name of this encoding method, which is known in the coding literature as *sequential plug-in* encoding. This encoding strategy has several advantages: 1) For common parameter estimators such as ML, this method can be shown to be universal; 2) Since all distribution parameters are fixed (pre-estimated) when encoding the $j$th sample, we can use the "original," nonuniversal distributions assumed for modeling $\mathbf{e}_j$ (LG) and $\mathbf{a}_j$ (Laplacian), which have closed forms and are usually faster to compute [together with (11)] than their universal mixture counterparts; 3) Furthermore, these original distributions are convex, so that in this case, given a fixed support, we are able to exactly minimize the codelength over the nonzero coefficient values; 4) With many samples available for parameter estimation, we can potentially afford more complex models.

*Residual:* We estimate $\theta_e$ in two steps. First, since the random variable $E$ is an independent sum of two random variables, $E = \hat{E} + N$, we have that $\mathrm{var}(E) = \mathrm{var}(\hat{E}) + \mathrm{var}(N) = \mathrm{var}(\hat{E}) + \sigma_e^2$. Now, since $\hat{E}$ is Laplacian, we have that $\mathrm{var}(\hat{E}) = 2\theta_e^2$. Combining both equations we have that

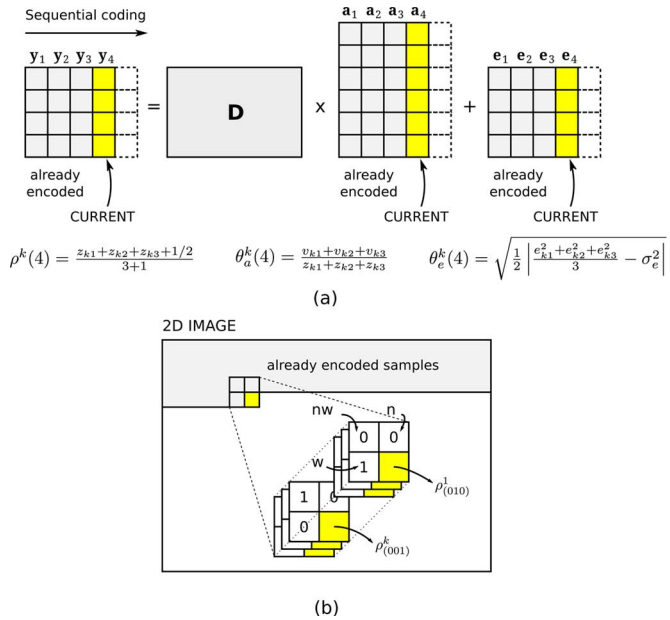


Fig. 4. Collaborative encoding scheme. (a) In this example, 3 samples have already been encoded, and we are about to encode sample 4. The formulas for estimating the various model parameters are shown for $j = 4$, in particular those for the error and the coefficients associated to the $k$th atom (the $k$th row of $\mathbf{A}$). (b) Markov model for the coefficients support matrix $\mathbf{Z}$. Here, a sample patch $\mathbf{y}$ is about to be encoded. Here the first atom was only used by the pixel to the west, so that the Markov state for modeling $z_1$ is $(n, w, nw) = (0, 1, 0)$, and $P(z_1 = 1) = \rho^1_{(0,1,0)}$. As for the $k$th atom, only the $nw$ pixel has used it, so that the Markov state for $z_k$ is (0,0,1), that is, $P(z_k = 1) = \rho^k_{(0,0,1)}$.

$\theta_e = 0.5\sqrt{\mathrm{var}(\hat{E}) - \sigma_e^2}$. With the noise variance $\sigma_e^2$ assumed known, and using the standard unbiased variance estimator, $\hat{\mathrm{var}}(\hat{E}) = (p(j-1))^{-1}\|\mathbf{E}_{[1,\ldots,(j-1)]}\|_F^2$, we obtain

$$\hat{\theta}_e = 0.5\sqrt{\max\{(p(j-1))^{-1}\|\mathbf{E}_{[1,\ldots,(j-1)]}\|_F^2 - \sigma_e^2, 0\}}$$

where the maximization guarantees that $\hat{\theta}_e \in \mathbb{R}^+$.

*Coefficients:* In the case of $\mathbf{a}$, we have in principle two unknown parameters, the probability of an element being nonzero, $\rho_a$, and the scale parameter of the Laplacian governing the nonzero values, $\theta_a$ (both previously handled with universal models). Here, however, we extend the model, drawing from the well known empirical fact that coefficients associated to different atoms can have very different statistics, both in frequency and variance. This is typical of DCT coefficients for example (see [40]), and has been consistently observed for learned dictionaries as well [20]. Therefore, we will consider a separate set of parameters $(\rho_a^k, \theta_a^k)$ for each *row* $k$ of $\mathbf{A}, \mathbf{a}^k$. We update such parameters from the coefficients observed in the respective row for the already-computed samples, $(a_{k1}, a_{k2}, \ldots, a_{k(j-1)})$, and encode each $k$th coefficient in $\mathbf{a}_j$ (more specifically, in $\mathbf{z}_j$, and $\mathbf{v}_j$), as the $j$th sample of the respective row. Concretely, let $n_1^k = \sum_{j'=1}^{(j-1)} z_{kj'}$ be the number of nonzero coefficients observed so far in the $k$th row. For $\rho_a^k$, we use the Krichevsky-Trofimov (KT) estimator [41]

$$\hat{\rho}_a^k = \frac{n_1^k + 0.5}{j} \tag{16}$$

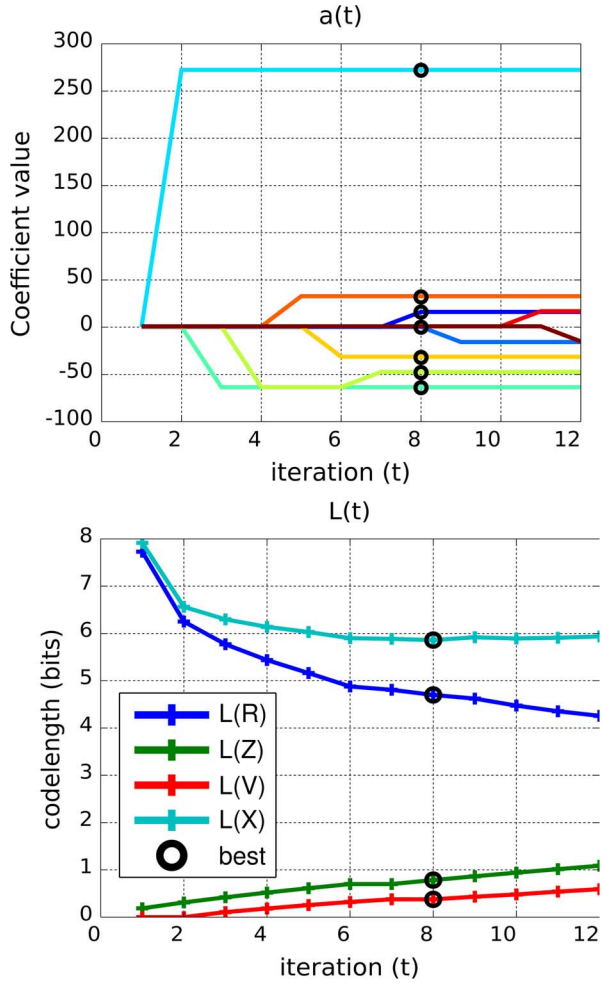


Fig. 5. Typical evolution of the COMPA algorithm. (a) coefficients. (b) codelength. The best iterate (code) is marked with a black circle. Also note that describing the support ($L(Z)$) actually takes more bits than describing the nonzero values ($L(V)$).

which is a universal plug-in encoding scheme for Bernoulli sequences of unknown parameter. For encoding $v_{kj}$, we apply the ML estimator for the exponential family to the nonzero coefficients observed so far in the $k$th row. Recalling that $v_{kj} = \max\{|a_{kj'}| - \delta_a, 0\}$, the resulting estimator is given by

$$\hat{\theta}_a^k = \frac{\sum_{j'=1}^{(j-1)} \max\{|a_{kj'}| - \delta_a, 0\}}{n_1^k}.$$

*Markovian Dependencies:* In many applications, spatially/temporally adjacent samples are statistically dependent. For example, we may assume that an atom is more likely to occur for a sample $j$ if it has been used by, say, the $(j-1)$th sample (see also [42]). In that case, we may consider different estimations of $\rho^k$ depending on the value of $z_{k(j-1)}$, $\rho_1^k = \mathrm{P}(z_{kj} = 1 | z_{k(j-1)} = 1)$, and $\rho_0^k = \mathrm{P}(z_{kj} = 1 \,|\, z_{k(j-1)} = 0)$. In particular, for the image processing results of Section VII, we use a Markovian model which depends on three previous samples, corresponding to the (causal) neighboring west, north, and northwest patches of the one being encoded. Thus, for each atom $k$ we will have 8 possible parameters, $\rho^k_{(n,w,nw)}, (n, w, nw) \in \{0, 1\}^3$, where each value of $(n, w, nw)$ indicates a possible *Markov state* in

which a sample may occur. This is depicted in Fig. 4(b). For each state $(n, w, nw)$, we estimate $\rho^k_{(n,w,nw)}$ using (16), with the average taken over the samples which occur in the same state $(n, w, nw)$.

## V. MDL Based Sparse Coding

For the encoding problem, $\mathbf{D}$ is fixed (it has already been learned), and we consider encoding a single data sample $\mathbf{y}$. The model selection problem here is that of choosing the model (indexed by the sparse code $\mathbf{a}$) among all the models belonging to the nested family of model classes $\mathcal{M}(\gamma) = \{\mathbf{a} \in \mathbb{R}^p, \|\mathbf{a}\|_0 \leq \gamma\}, \gamma = 0, \ldots, p$, that yields the smallest codelength for describing $\mathbf{y}$. In principle, this calls for finding the best model $\mathbf{a}(\gamma)$ within each model class $\mathcal{M}(\gamma)$, and then selecting $\hat{\mathbf{a}} = \arg\min_{0 \leq \gamma \leq p} L(\mathbf{y}, \mathbf{a}(\gamma))$. However, in order to be computationally efficient, and as with most sparse coding and model selection algorithms, several simplifications and approximations are needed. Let us first consider the problem of finding $\mathbf{a}(\gamma)$

$$\begin{aligned}\mathbf{a}(\gamma) &:= \arg \min_{\mathbf{a} \in \mathcal{M}(\gamma)} L(\mathbf{y}, \mathbf{a}) \\ &= \arg \min_{\mathbf{a} \in \mathbb{R}^p} -\log \mathrm{P}_E(\mathbf{y} - \mathbf{D}\mathbf{a}) - \log \mathrm{P}(\mathbf{z}) - \log \mathrm{P}(\mathbf{s}|\mathbf{z}) \\ &\quad - \log \mathrm{P}_V(\mathbf{a}|\mathbf{s}, \mathbf{z}) \\ &= \arg \min_{\mathbf{a} \in \mathbb{R}^p} -\log \mathrm{P}_E(\mathbf{y} - \mathbf{D}\mathbf{a}) \\ &\quad - \log \binom{p}{\|\mathbf{a}\|_0} + \|\mathbf{a}\|_0 \\ &\quad - \log \mathrm{P}_V(\mathbf{a}) \quad \text{s.t. } \|\mathbf{a}\|_0 \leq \gamma. \end{aligned} \tag{17}$$

For quantized $\mathbf{a}$, this is an optimization problem over a discrete, infinite domain, with a nonconvex (in the continuous domain) constraint, and a nondifferentiable cost function in $\mathbf{a}$. Based on the literature on sparse coding, at least two alternatives can be devised at this point. One way is to use a pursuit technique, e.g., [22]. Another option is to use a convex relaxation of the codelength function, e.g., [43]. For the sake of brevity, here we will describe an algorithm loosely based on the first alternative. Details on the convex relaxation method for MDL-based sparse coding will be published elsewhere.

The pursuit-like algorithm, which we call COdelength-Minimizing Pursuit Algorithm (COMPA), is summarized in Algorithm 1. This is a nongreedy crossbreed between matching pursuit (MP) and forward stepwise selection (FSS) [34]. As with those methods, COMPA starts with the empty solution $\mathbf{a}^{(0)} = \mathbf{0}$, and updates the value of one single coefficient at each iteration. Then, given the current correlation $\mathbf{g}^{(t)} = \mathbf{D}\mathbf{e}^{(t)}$ between the dictionary atoms and the current residual, each $k$th coefficient in $\mathbf{a}^{(t)}$ is tentatively incremented (or decremented) by $\Delta_k = [g_k^{(t)}]$, and a candidate codelength $\hat{L}_k$ is computed in each case. The coefficient that produces the smallest $\hat{L}(\mathbf{y}, \mathbf{a})$ is updated to produce $\mathbf{a}^{(t+1)}$.

The logic behind this procedure is that the codelength cost of adding a new coefficient to the support is usually very high, so that adding a new coefficient only makes sense if its contribution is high enough to produce some noticeable effect in the other parts of the codelength. A variant of this algorithm was also implemented where, for each candidate $k$, the value of the increment $\Delta_k$ was refined in order to minimize $\hat{L}_k$. However, this variant turned out to be significantly slower, and the compression gains where below 0.01 bits per sample (uncompressed codelength is 8 bits per sample). Assuming that $L^{(t)}$ is unimodal, the algorithm stops if the codelength of a new iterate is larger than the previous one. To assess the validity of this assumption, we also implemented a variant which stops, as MP or FSS, when the residual-coefficients correlation $\|\mathbf{g}^{(t)}\|_\infty$ is no longer significant, which typically requires many more iterations. With this variant we obtained a negligible improvement of 0.004 bits per sample, while increasing the computational cost about three times due to the extra iterations required.

**Algorithm 1**: COdelength Minimizing Pursuit Algorithm (COMPA)

**Input**: Data sample $\mathbf{y}$, dictionary $\mathbf{D}$
**Output**: $\hat{\mathbf{a}}$, $\hat{\mathbf{e}}$
**initialize** $t \leftarrow 0; \mathbf{a}^{(0)} \leftarrow \mathbf{0}; \mathbf{e} \leftarrow \mathbf{y}; L^{(0)} \leftarrow L(\mathbf{y}, \mathbf{0}); \mathbf{g}^{(t)} \leftarrow \mathbf{D}^T\mathbf{e}^{(t)}$ ;
*// $\mathbf{g}^{(t)}$ correlation of current residual with the dictionary*
**repeat**
  **for** $k \leftarrow 1, 2, \ldots, p$ **do**
    $\Delta_k \leftarrow [g_k^{(t)}]_{\delta_a}$ ; *// step $\Delta_k$ is correlation, quantized to prec. $\delta_a$*
    $\tilde{L}_k \leftarrow L([\mathbf{e} - \Delta_k \mathbf{d}_k]_{\delta_e}, \mathbf{a} + \Delta_k \omega_k)$ ; *// $\omega_k$ = k-th canonical vec. of $\mathbb{R}^p$*
  **end**
  $L^{(t+1)} \leftarrow \min\{\tilde{L}_k : k = 1, \ldots, p\}$ ;
  $\mathbf{a}^{(t+1)} \leftarrow \mathbf{a}^{(t)} + \Delta_{\hat{k}} \omega_{\hat{k}}$ ; *// update coefficients vector*
  $\mathbf{g}^{(t+1)} \leftarrow \mathbf{g}^{(t)} - \Delta_{\hat{k}} \mathbf{d}_{\hat{k}}$ ; *// update correlation*
  $t \leftarrow t + 1$ ;
**until** $L^{(t)} \geq L^{(t-1)}$ ;
$\hat{\mathbf{a}} \leftarrow \mathbf{a}(\gamma - 1)$ ;
$\hat{\mathbf{e}} \leftarrow [\mathbf{y} - \mathbf{D}\hat{\mathbf{a}}]$ ;
STOP ;

## VI. MDL-Based Dictionary Learning

Given that our sparse coding algorithm in Section V can select the best support size $\gamma$ for each sample in $\mathbf{Y}$, the definition of the model class $\mathcal{M}(\gamma, p)$ given in Section III, which assumes the same $\gamma$ for all samples in $\mathbf{Y}$, is no longer appropriate (we could of course add 0-weight coefficients to make $\gamma$ equal for all data). Instead, for dictionary learning, we consider the model class family $\mathcal{M}(p) = \{(\mathbf{A}, \mathbf{D}), \mathbf{D} \in \mathbb{R}^{m \times p}, \mathbf{a}_j \in \mathcal{M}(\gamma; \mathbf{D}), j = 1, \ldots, n\}$, where $\mathcal{M}(\gamma; \mathbf{D})$ is the model class family of sparse codes based on a fixed dictionary $\mathbf{D}$ defined in Section V, with the dependency on $\mathbf{D}$ made explicit. It is easy to see that the model classes $\mathcal{M}(p)$ are nested. We now need to solve

$$(\mathbf{A}(p), \mathbf{D}(p)) = \arg \min_{(\mathbf{A}, \mathbf{D}) \in \mathcal{M}(p)} L(\mathbf{E}, \mathbf{A}, \mathbf{D}) \tag{18}$$

for $p = 0, 1, \ldots$, and then choose $(\hat{\mathbf{A}}, \hat{\mathbf{D}}) = (\mathbf{A}(\hat{p}), \mathbf{D}(\hat{p}))$ with the optimal dictionary size

$$\hat{p} = \arg \min_p \{L(\mathbf{E}, \mathbf{A}(p), \mathbf{D}(p)) : p = 0, 1, \ldots\}.$$

As with sparse coding, here we exploit the nested nature of the model classes to speed up the model selection. For this, we propose a forward-selection algorithm, described in Algorithm 2, which starts from $\mathcal{M}(0)$ (the empty dictionary), and then approximates the best model in $\mathcal{M}(p+1)$ by adding a new atom to the dictionary computed for $\mathcal{M}(p)$ and then invoking Algorithm 3, which is discussed in depth in the next subsection.

**Algorithm 2**: MDL-based dictionary learning via forward selection.

**Input**: Data $\mathbf{Y}$
**Output**: $(\hat{\mathbf{A}}, \hat{\mathbf{D}})$
**initialize** $p \leftarrow 0$; $\mathbf{A}(0) \leftarrow \emptyset$; $\mathbf{D}(0) \leftarrow \emptyset$; $\mathbf{E}(0) \leftarrow \mathbf{Y}$;
$L(0) \leftarrow L(\mathbf{E}(0), \mathbf{A}(0), \mathbf{D}(0))$ ;
**repeat**
  $\tilde{\mathbf{d}} \leftarrow \mathbf{u}_1, \mathbf{U}\Sigma\mathbf{V}^\intercal = \mathbf{E}(p)$ // *Initial value of new atom is the left-eigenvector associated to the largest singular value of* $\mathbf{E}^{(t)}$.
  $\mathbf{D}^0 \leftarrow [\mathbf{D}(p) \,|\, \tilde{\mathbf{d}}]$ // *Initial dictionary for optimization below.*
  $(\mathbf{A}(p+1), \mathbf{D}(p+1)) \leftarrow \arg\min_{(\mathbf{A},\mathbf{D}) \in \mathcal{M}(p+1)} L(\mathbf{E}, \mathbf{A}, \mathbf{D})$ // *Optimize dict. via Algorithm 3*
  $p \leftarrow p + 1$ ;
  $L(p) \leftarrow L(\mathbf{E}(p), \mathbf{A}(p), \mathbf{D}(p))$ ;
**until** $L(p) \geq L(p-1)$ ;
$\hat{\mathbf{A}} \leftarrow \mathbf{A}(p-1)$; $\hat{\mathbf{D}} \leftarrow \mathbf{D}(p-1)$;

**Algorithm 3**: MDL-based dictionary learning for a given size $p$

**Input**: Data $\mathbf{Y}$, initial dictionary $\mathbf{D}^0$, multiplier $\lambda$, $\eta$
**Output**: Local-optimum $(\hat{\mathbf{A}}, \hat{\mathbf{D}})$
**initialize** $\mathbf{D}^{(0)} = \mathbf{D}^0$, $t = 1$ ;
**repeat**
  **for** $j = 1, \ldots, n$ **do**
    $\mathbf{a}_j^{(t)} \leftarrow \arg\min_{\mathbf{A}} L(\mathbf{e}, \mathbf{a}, \mathbf{D}^{(t-1)})$ ;
  **end**
  Update plug-in parameters: $\theta_e$, $\left\{(\theta_a^k, \rho_a^k), k = 1, \ldots, p\right\}$, $\theta_d$;
  $\mathbf{D}^{(t)} \leftarrow \arg\min_{\mathbf{D}} L(\mathbf{E}, \mathbf{A}, \mathbf{D})$ ;
  $t \leftarrow t + 1$ ;
**until** $\frac{\|\mathbf{D}^{(t)} - \mathbf{D}^{(t-1)}\|_2}{\|\mathbf{D}^{(t)}\|_2} \leq \epsilon$ ;

A backward-selection algorithm was also developed which first learns the model for $\mathcal{M}(p_{\max})$ via (3), where $p_{\max}$ is a given maximum dictionary size, and then prunes the less frequently used atoms until no further decrease in codelength is observed. This algorithm allows us to provide especially constructed initial dictionaries for Algorithm (3), e.g., an (overcomplete) DCT frame, which can be critical for finding good local minima of the nonconvex problem (18). We do this for example to learn a dictionary for the whole class of natural images, see Section VII.

### A. Optimizing the Dictionary for Fixed $p$

For fixed $p$, and given an initial $\mathbf{D}$, Algorithm 3 adapts the atoms of $\mathbf{D}$ to fit the training data $\mathbf{Y}$. At the high level, our algorithm is very similar to the traditional approach of alternate minimization over $(\mathbf{A}, \mathbf{D})$. However, there are a number of important differences, namely: 1) The cost function minimized is now the cumulative codelength of describing $\mathbf{Y}, L(\mathbf{E}, \mathbf{A}, \mathbf{D})$; 2) Minimizing over $\mathbf{A}$ is done sample by sample following Section V; 3) Since $\mathbf{D}$ needs to be described as well, it has an associated codelength (see Section IV-C), resulting in regularized dictionary update, described below; 4) in a crossbreed between expectation-maximization, and plug-in estimation, we estimate the model parameters for the current iterate $(\mathbf{E}^{(t)}, \mathbf{A}^{(t)}, \mathbf{D}^{(t)})$, from the accumulated statistics of previous iterates $\{(\mathbf{E}^{(t')}\mathbf{A}^{(t')}, \mathbf{D}^{(t')}), t' = 1, \ldots, t-1\}$. At the end of the learning process, these parameters are "saved" as part of the learned model and can be used for modeling future data along with $\mathbf{D}$.

At the $t$th iteration of the alternate minimization between $\mathbf{D}$ and $\mathbf{A}$, with $\mathbf{A}^{(t)}$ just computed and kept fixed, the dictionary step consists of solving the subproblem

$$\begin{aligned}\mathbf{D}^{(t)} &= \arg\min_{\mathbf{D} \in \mathbb{R}^{m \times p}} L(\mathbf{Y}, \mathbf{A}^{(t)}, \mathbf{D}) \\ &= \arg\min_{\mathbf{D} \in \mathbb{R}^{m \times p}} L(\mathbf{Y}|\mathbf{A}^{(t)}, \mathbf{D}) + L(\mathbf{D}).\end{aligned}$$

According to Section IV-C, we have $L(\mathbf{D}) = \frac{1}{\theta_d^{(t)}} \sum_{k=1}^p \|\mathbf{W}\mathbf{d}_k\|_1$, where $\theta_d^{(t)} = \frac{1}{mp} \sum_{k=1}^p \sum_{i=1}^m |d_{ik}^{(t-1)}|$ is the Laplacian MLE of $\theta_d$ based on $\mathbf{D}^{(t-1)}$. Correspondingly, the data fitting term, via (11) and disregarding the constant terms, is given by $L(\mathbf{Y}|\mathbf{A}^{(t)}, \mathbf{D}) = L(\mathbf{Y} - \mathbf{D}\mathbf{A}^{(t)}|\theta_e^{(t)}, \sigma_e^2) = \sum_{j=1}^n \sum_{i=1}^m -\log LG(y_{ij} - (\mathbf{D}\mathbf{A}^{(t)})_{ij}; \theta_e^{(t)}, \sigma_e^2)$, where $\theta_e^{(t)}$ is the estimator of $\theta_e$ given $\mathbf{E} = \mathbf{Y} - \mathbf{D}^{(t-1)}\mathbf{A}^{(t)}$ (see Section IV-D) and $\sigma_e^2$ is assumed known. The problem can now be written as,

$$\mathbf{D}^{(t)} = \arg\min_{\mathbf{D}} L\left(\mathbf{Y} - \mathbf{D}\mathbf{A}^{(t)}|\theta_e^{(t)}, \sigma_e^2\right) + \theta_d^{(t)} \sum_{k=1}^p \|\mathbf{W}\mathbf{d}_k\|_1. \tag{19}$$

For general $\mathbf{W}$, the optimization of (19) is challenging since none of the above terms are separable, in particular, the nondifferentiable $\ell_1$ term. However, since $\mathbf{W}$ is easily invertible (as described in Section IV-C, it is lower triangular with 1's in the diagonal), we can perform a change of variables and solve the equivalent problem in the *prediction residual matrix* $\mathbf{U} = \mathbf{W}\mathbf{D}$ instead,

$$\hat{\mathbf{U}} = \arg\min_{\mathbf{U}} L\left(\mathbf{Y} - \mathbf{W}^{-1}\mathbf{U}\mathbf{A}^{(t)}|\theta_e^{(t)}, \sigma_e^2\right) + \theta_d^{(t)} \sum_{k=1}^p \|\mathbf{u}_k\|_1. \tag{20}$$

Since the regularization term in (20) is decoupled in the elements of $\mathbf{U}$, and $L(\mathbf{Y} - \mathbf{W}^{-1}\mathbf{U}\mathbf{A}|\theta_e^{(t)}, \sigma_e^2)$ is convex and differentiable in $\mathbf{U}$ (see Fig. 2(a)), (20) can be efficiently solved using the existing techniques for separable nondifferentiable regularization terms. In our case, we employ the backtracking variant of FISTA [27], focusing on an efficient numerical evaluation of each step.

## VII. Experimental Results

### A. Coding Performance

The first experiment in this section assesses the ability of our coding scheme to actually produce a compressed description of the data, in this case 8-bit gray-scale images. To this end, a dictionary $\mathbf{D}$ was learned using the backward-selection algorithm, for the training samples from the Pascal'06 image database,[7] converted to 8-bit gray-scale images and decomposed into $8 \times 8$ patches. The initial dictionary was an overcomplete DCT frame with $p = 256$. The resulting global dictionary $\mathbf{D}$ has $p = 250$ atoms. We then encoded the testing samples from the same database, obtaining an average codelength of 4.1 bits

[7]http://pascallin.ecs.soton.ac.uk/challenges/VOC/databases.html.

per pixel (bpp), confirming the ability of our model to produce a compressed description of the data.

### B. Learning Performance

We compare the performance of the forward and backward dictionary learning algorithms proposed in Section VI by applying each method to learn a dictionary for the standard "Boats" image (taken from the SIPI database,[8] along with "Lena," "Barbara," and "Peppers" used in the following experiments), and then measuring the final codelength and computation time. For the backward case, the initial dictionary is the global dictionary learned in the previous experiment. As for the forward method, we also include a faster "partial update" variant which performs a few (10) iterations of Algorithm 3 after adding a new atom, instead of allowing it to converge. The results were a compression level of 5.13 bpp at a computational cost of 3900 s for the backward method, 5.19 bpp requiring 800 s for the convergent forward method, and 5.22 bpp requiring 150 s for the partial forward method (the running times were measured for a parallelized C++ implementation running on an Athlon Phenom II X6 at 2.6 GHz). In summary, all three methods reach similar, significant, compression levels. Slightly better results are obtained with the backward method, at the cost of a significant increase in computational time. On the other hand, the partial forward variant is significantly faster than the other two, yielding similar codelengths.

### C. Denoising of Natural Images

The task in this case is to estimate a clean image from an observed noisy version whose pixels are corrupted by AWGN of known variance $\sigma_e^2$. Here $\mathbf{Y}$ contains all (overlapping) $8 \times 8$ patches from the noisy image. The denoising algorithm proceeds in two stages. In the first one, a dictionary $\mathbf{D}$ is learned from the noisy image patches $\mathbf{Y}$. We use the backward selection algorithm since it allows us to use the global dictionary as the starting point, a common practice in this type of problems, [29], [31]. Second, the clean patches are estimated as sparse combinations of atoms from $\mathbf{D}$. In our case, the second stage admits two variants. The first one is a rate-distortion (RD) procedure akin to the traditional method used for example in [29], where each clean sample $\hat{\mathbf{y}}_j$ is estimated using a distortion-constrained formulation. In our case, we minimize the codelength (or "rate") of describing $\mathbf{y}_j$ up to a prescribed distortion proportional to the noise level, $\hat{\mathbf{y}}_j = \mathbf{D}\hat{\mathbf{a}}_j, \hat{\mathbf{a}}_j = \arg\min_{\mathbf{u}} L(\mathbf{u})$ s.t. $\|\mathbf{y}_j - \mathbf{D}\mathbf{u}\|_2 \leq C\sigma_N^2$. Here we use $C = 1.0$. The second variant, coined "postthresholding" (PT) is more consistent with the learning phase, and is truly parameter-free, since the estimation derives from the same codelength minimization procedure used for learning the dictionary $\mathbf{D}$. In this case we obtain an initial estimate $\tilde{\mathbf{y}}_j = \mathbf{D}\tilde{\mathbf{a}}_j$, $\tilde{\mathbf{a}}_j = \arg\min_{\mathbf{u}} L(\mathbf{u}) + L(\mathbf{y}_j \,|\, \mathbf{u})$. However, according to the model developed in Section IV-B, the encoding residual $\tilde{\mathbf{e}} = \mathbf{y}_j - \tilde{\mathbf{y}}_j$ may contain a significant portion of clean data due to modeling errors. We can then think of $\tilde{\mathbf{e}}$ as clean data corrupted by noise of variance $\sigma_e^2$. To extract the clean portion,

[8]http://sipi.usc.edu/database/database.php?volume=misc&image=38#top.

| $\sigma_e = 10$ | PT | RD | [17] | [29] | [4] | $\sigma_e = 20$ | PT | RD | [17] | [29] |
|---|---|---|---|---|---|---|---|---|---|---|
| lena | 34.9 | 35.2 | 32.4 | 35.5 | 35.6 | 32.0 | 32.2 | 29.4 | 32.4 | 32.7 |
| barbara | 33.0 | 33.8 | 29.4 | 34.4 | 34.6 | 29.7 | 30.6 | 25.7 | 30.8 | 31.1 |
| boat | 33.1 | 33.2 | 30.5 | 33.6 | 33.7 | 29.5 | 30.3 | 27.5 | 30.3 | 30.7 |
| peppers | 34.1 | 34.4 | 32.2 | 34.3 | 34.8 | 31.7 | 31.6 | 29.4 | 30.8 | 32.4 |

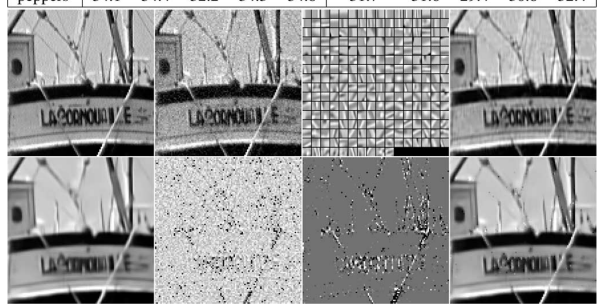

Fig. 6. Denoising results. Table: denoising performance, in PSNR, of K-SVD [29], MDL denoising [17], and the PT and RD denoising variants. Images, top row: clean "Boats," noisy version, learned dictionary for this image (final $p = 248$), image recovered using RD. Images, bottom row: image reconstructed from the initial estimation $\tilde{\mathbf{y}}_j$ obtained in the PT method, its residual, portion of residual that was added back, final PT estimation.

we solve another codelength-minimization subproblem, this time with a Gaussian prior for the error, and a Laplacian prior for the clean part, $\bar{\mathbf{e}}_j = \arg\min_u \frac{1}{\sigma_e^2}\|\tilde{\mathbf{e}}_j - \mathbf{u}\|_2 + \frac{1}{\hat{\theta}_e}\|\mathbf{u}\|_1$, where $\hat{\theta}_{\mathbf{e}} = \sqrt{0.5\max\{0, \mathrm{var}(\tilde{\mathbf{e}}_j) - \sigma_e^2\}}$, following Section IV-D. We then compute the final estimate as $\hat{\mathbf{y}}_j = \tilde{\mathbf{y}}_j + \bar{\mathbf{e}}_j$. In either variant, the model used for $L(\mathbf{a})$ includes the Markovian dependency between the occurrence of each atom in a patch and its previously encoded neighbors, as described in Section IV-D.

Denoising performance is summarized in Fig. 6, along with a detail of the result for $\sigma_e = 10$ for the "Boats" image in Fig. 6. In all cases, there is a 1 to 5 dB improvement over the best MDL-based results in [17], thus showing the relevance of overcoming the limitations in previous MDL applications to sparse coding. Both the RD and PT methods yield results which are comparable to those of [29], which depend significantly on several carefully tuned parameters. The Bayesian approach [4] produces slightly better results in all cases, but the additional computational cost is very significant. While our method requires about one minute to produce the results of Fig. 6, the implementation available from the authors of [4] requires over an hour on the same hardware in all cases.[9] While the RD variant performs better than PT in terms of PSNR, PT is faster and tends to produce less artifacts than RD, thus resulting in more visually pleasant images than RD. This, which can be clearly seen in Fig. 6, occurs in all other cases as well. Including the Markov dependency in $L(\mathbf{a})$ produced an average improvement of up to 0.2 dB.

### D. Texture Mosaic Segmentation

Here we are given $c$ images with sample textures, and a target mosaic of textures,[10] and the task is to assign each pixel in the mosaic to one of the textures. This classic computer vision problem has been addressed using traditional sparse models for example in [32] and in our own previous work [33]. Here again,

[9]To the best of our knowledge, these results, as well as those in [4], [29], are among the best that can be obtained for gray-scale images without using multi-scale and/or spatial aggregation of patches as in [44], [45].

[10]Taken from http://www.ux.uis.no/~tranden/.

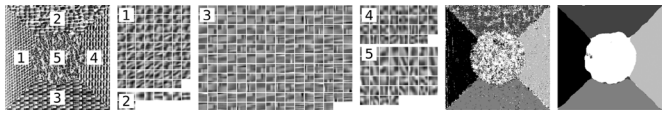

Fig. 7. Left to right: Texture mosaic, dictionaries learned for each class (note the automatically learned different sizes), patch-wise codelength-based classification map—each shade of gray corresponds to a texture class—(77.0% success rate), classification map obtained by averaging the codelength over a neighborhood of patches (95.4% success rate).

the images are decomposed into overlapping patches. This time a different dictionary is learned for each texture using patches from corresponding training images. In order to capture the texture patterns, a patch width $w = 16$ was used. Then, each patch in the mosaic is encoded using all available dictionaries, and its center pixel is assigned to the class which produced the shortest description length for that patch.

This seemingly natural procedure results in a success rate of 77%, which is consistent with the second picture of Fig. 7. The problem is that this procedure is inconsistent with the learning formulation, because each dictionary is adapted to minimize the *average* codelength of describing each patch in the respective texture. Therefore, good results can only be expected if the decision is made for groups of patches simultaneously, that is, by considering the cumulative codelength of a set of patches. We implement this by deciding on each patch on the basis of comparing the average codelength obtained with each dictionary for encoding that patch and all patches in a circular neighborhood with a radius of 20 pixels. The success rate in this case is 95.3%, which is comparable to the state-of-the art for this type of problems (see for example [32], which learns sparse models for explicitly maximizing the success rate). The Markovian model improved our results by 1%.

### *E. Low-Rank Matrix Approximation*

The low-rank matrix approximation family of problems (see [46] for a review) can be seen as an extension to the problem of sparse coding where sparsity is substituted by matrix rank. Concretely, the task is to recover a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ from an incomplete and/or corrupted observation $\mathbf{Y}$, under the assumption that the rank of $\mathbf{A}$, $\mathrm{rank}(\mathbf{A})$, is small. As with sparse coding, $\mathrm{rank}(\mathbf{A})$ is relaxed using the $\ell_1$ equivalent for matrix rank, which is the nuclear norm, $\|\mathbf{A}\|_* := \sum_i \sigma_i(\mathbf{A})$, where $\sigma_i(\mathbf{A})$ is the $i$th singular value of $\mathbf{A}$. It has been shown in [46] that, under certain assumptions on $\mathrm{rank}(\mathbf{A})$, the following estimation function is able to recover $\mathbf{A}$ from a noisy observation $\mathbf{Y}$, and with a significant fraction of its coefficients arbitrarily corrupted

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{W}} \|\mathbf{W}\|_* + \lambda\|\mathbf{Y}-\mathbf{W}\|_1 \qquad \lambda = 1/\sqrt{\max\{m,n\}}. \tag{21}$$

A common proof of concept is to use this framework for robust background estimation in camera surveillance video sequences [47], and we apply our proposed framework for the same application.

To perform our MDL-based model selection within this formulation, we solve (21) for increasing values of $\lambda$, obtaining a low-rank approximation to $\mathbf{A}$, $(\mathbf{A}(\lambda), \mathbf{E}(\lambda) = \mathbf{Y} - \mathbf{A}(\lambda))$, which we encode using the universal models described in Section IV. We modified the algorithm described in [48] to allow for warm restarts, using the solution for the previous $\lambda$ as a starting point for the next $\lambda$ for faster convergence.

Consistently with the $\ell_1$ fitting term of (21), we encode the nonzero values of $\mathbf{E}(\lambda)$ as a Laplacian sequence of unknown parameter. To exploit the potential sparsity in $\mathbf{E}(\lambda)$, the locations of the nonzero values are encoded, as in Section IV-A, using an enumerative two-parts code for Bernoulli sequences of unknown parameter. To exploit low-rank in the encoding, the matrix $\mathbf{A}(\lambda)$ is encoded via its reduced SVD decomposition $\mathbf{A}(\lambda) = \mathbf{U}(\lambda)\Sigma(\lambda)\mathbf{V}(\lambda)^\intercal$. For $\mathrm{rank}(\mathbf{A}(\lambda)) = r$, we have that $\mathbf{U}(\lambda) \in \mathbb{R}^{m\times r}$ are the left-eigenvectors, $\Sigma \in \mathbb{R}^{r\times r}$ is the diagonal matrix whose diagonal are the nonzero singular values of $\mathbf{A}(\lambda)$, and $\mathbf{V}(\lambda) \in \mathbb{R}^{r\times n}$ are the right-eigenvectors of $\mathbf{A}(\lambda)$. Each column of $\mathbf{U}$ is encoded (in this video example) as a smooth image via a causal bilinear predictor identical to the one used for predictive coding of $\mathbf{D}$ in IV.C, using a Laplacian model for the prediction residuals. Each column of $\mathbf{V}$ is encoded as a smooth one-dimensional sequence, using a zero order predictor (the predicted value for the next coefficient is the previous coefficient value), with a Laplacian prior on the prediction residuals. Finally, the values of $\mathbf{\Sigma}$, which can be arbitrary, are quantized and encoded using the universal code for integers [49].

The encoding method is very simple, with all unknown parameters encoded using a two-parts code, and codelenghts for the discretized Laplacian precomputed in look-up tables. Quantization for this case is as follows: the codelength associated with the $r$ nonzero singular values is negligible, and we minimize unwanted distortion encoding them with high precision $(1e-16)$. As for the columns of $\mathbf{U}$ and $\mathbf{V}$, they all have unit norm, so that the average magnitude of their elements are close to $\sqrt{1/m}$ and $\sqrt{1/n}$, respectively. Based on this, our algorithm encodes the data with $\delta_u = Q/\sqrt{m}$ as the precision for encoding $\mathbf{U}$, and $\delta_v = Q/\sqrt{m}$ for $\mathbf{V}$, for several values of $Q$ in (0,1), keeping the one producing the smallest codelength. The MDL-based estimation algorithm then chooses the model for which the codelength $L(\mathbf{Y};\lambda) = L(\mathbf{U}(\lambda)) + L(\Sigma(\lambda)) + L(\mathbf{V}(\lambda))$ is minimized.

As in [47], here we show results for two sequences taken from [50]: "Lobby" [Fig. 8(a)], and "ShoppingMall" [Fig. 8(b)]. Full videos can be viewed at http://www.tc.umn.edu/nacho/lowrank/.

In both cases, the recovered backgrounds are very accurate. In particular, for the Lobby sequence, the selected model captures just the eigenvectors needed to recover the background along with its lighting changes, including corrections for local shadows, leaving out only the people passing by.

## VIII. Concluding Remarks

We have presented an MDL-based sparse modeling framework, which automatically adapts to the inherent complexity of the data at hand using codelength as a metric.

The framework features a sparse coding algorithm and automatic tuning of the sparsity level on a per-sample basis, including a sequential collaborative variant which adapts

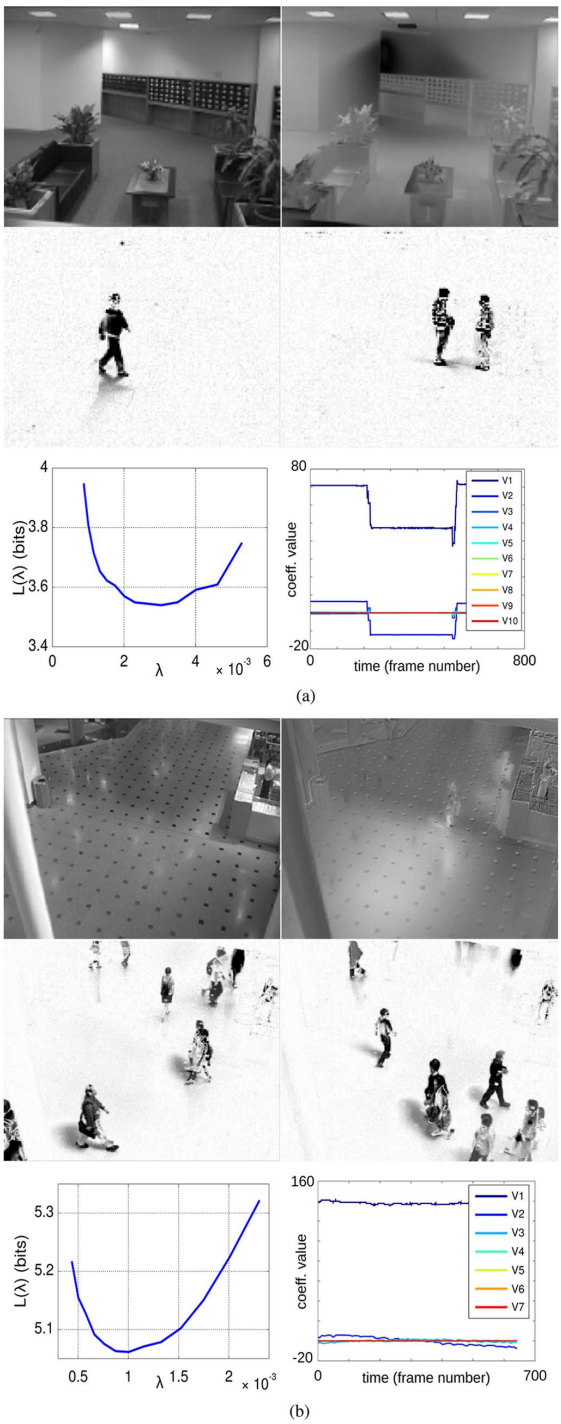


Fig. 8. Low-rank approximation results. Both figures show the first two left-eigenvectors as 2D images at the top, two sample frames from the approximation error sequences in the middle, which should contain the people that were removed from the videos, and the curve $L(\lambda)$ and the right-eigenvalues, scaled by $\Sigma$ (representing the "activity" of each left-eigenvector along time), at the bottom. (a) Results for "Lobby" sequence, featuring a room with lights that are switched off and on. The rank of the approximation for this case is $\mathrm{rank} = 10$. The moment where the lights are turned off is clearly seen here as the "square pulse" in the middle of the first two right-eigenvectors (bottom-right). Also note how $\mathbf{u}_2$ (top-right) compensates for changes in shadows. (b) Results for "ShoppingMall," a fixed camera looking at a crowded hall. In this case, the rank of the approximation decomposition is $\mathrm{rank} = 7$. Here, the first left-eigenvector models the background, whereas the rest tend to capture people that stood still for a while. Here we see the "phantom" of two such persons in the second left-eigenvector (top-right).

the model parameters as it processes new samples, and two dictionary learning variants which learn the size of the dictionaries from the data. In all cases, the information-theoretic formulation led to robust coding and learning formulations, including novel robust metrics for the fitting term (LG and MOEG), and robust $\ell_1$-based dictionary regularization term. This formulation also allowed us to easily incorporate more prior information into the coding/learning process, such as Markovian spatial dependencies, by means of simple modifications to the probability models used.

As a result, the framework can be applied out-of-the-box to very different applications, from image denoising to low-rank matrix approximation, obtaining competitive results in all the cases presented, with minimal interaction from the user.

## REFERENCES

[1] A. Bruckstein, D. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Rev.*, vol. 51, no. 1, pp. 34–81, Feb. 2009.

[2] R. Rubinstein, A. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proc. IEEE*, vol. 98, no. 6, pp. 1045–1057, Jun. 2010.

[3] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proc. IEEE*, vol. 98, no. 6, pp. 1031–1044, Jun. 2010.

[4] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin, "Nonparametric Bayesian dictionary learning for analysis of noisy and incomplete images," *IEEE Trans. Image Process.*, 2011, submitted for publication.

[5] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Autom. Contr.*, vol. 19, pp. 716–723, 1974.

[6] G. Schwartz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, 1978.

[7] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

[8] J. Rissanen, "Universal coding, information, prediction, and estimation," *IEEE Trans. Inf. Theory*, vol. 30, no. 4, pp. 629–636, 1984.

[9] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2743–2760, 1998.

[10] T. Cover and J. Thomas, *Elements of Information Theory*, 2nd ed. New York: Wiley, 2006.

[11] S. Kritchman and B. Nadler, "Non-parametric detection of the number of signals: Hypothesis testing and random matrix theory," *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 3930–3941, Oct. 2009.

[12] B. Nadler, "Nonparametric detection of signals by information theoretic criteria: Performance analysis and an improved estimator," *IEEE Trans. Signal Process.*, vol. 58, no. 5, pp. 2746–2756, May 2010.

[13] N. Saito, "Simultaneous noise suppression and signal compression using a library of orthonormal bases and the MDL criterion," in *Wavelets in Geophysics*, E. Foufoula-Georgiou and P. Kumar, Eds. New York: Academic, 1994, pp. 299–324.

[14] H. Krim and J.-C. Pesquet, On the Statistics of Best Bases Criteria MIT LIDS, Tech. Rep., 1995.

[15] P. Moulin and J. Liu, "Analysis of multiresolution image denoising schemes using generalized-Gaussian and complexity priors," *IEEE Trans. Inf. Theory*, Apr. 1999.

[16] J. Rissanen, "MDL denoising," *IEEE Trans. Inf. Theory*, vol. 46, no. 7, pp. 2537–2543, 2000.

[17] T. Roos, P. Myllymäki, and J. Rissanen, "MDL denoising revisited," *IEEE Trans. Signal Process.*, vol. 57, no. 9, pp. 3347–3360, 2009.

[18] I. Ramírez and G. Sapiro, "Sparse coding and dictionary learning based on the MDL principle," in *Proc. ICASSP 2011*, Prague, Czech. Republic, May 2011.

[19] I. Ramírez and G. Sapiro, "Low-rank data modeling via the minimum description length principle," in *Proc. ICASSP 2012*, 2011, arXiv:1109.6297v1, to be published.

[20] I. Ramírez and G. Sapiro, "Universal regularizers for robust sparse coding and modeling," Aug. 2010, Preprint available in arXiv:1003.2941v2 [cs.IT].

[21] P. J. Huber, "Robust estimation of a location parameter," *Ann. Statist.*, vol. 53, pp. 73–101, 1964.

[22] S. Mallat and Z. Zhang, "Matching pursuit in a time-frequency dictionary," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, 1993.
[23] R. Tibshirani, "Regression shrinkage and selection via the LASSO," *J. Royal Statist. Soc.: Ser. B*, vol. 58, no. 1, pp. 267–288, 1996.
[24] E. J. Candès, "Compressive sampling," in *Proc. Int. Congress of Math.*, Aug. 2006, vol. 3.
[25] D. L. Donoho and I. M. Johnston, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, Sep. 1994.
[26] J. Friedman, H. Hastie, and R. Tibshirani, "Regularized paths for generalized linear models via coordinate descent," *J. Stat. Soft.*, vol. 33, no. 1, 2008.
[27] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
[28] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Statist.*, vol. 32, no. 2, pp. 407–499, 2004.
[29] M. Aharon, M. Elad, and A. Bruckstein, "The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.
[30] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, 2010.
[31] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Trans. Image Process.*, vol. 17, no. 1, pp. 53–69, Jan. 2008.
[32] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis," in *Proc. IEEE CVPR*, Anchorage, AK, Jun. 2008.
[33] I. Ramírez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," in *Proc. CVPR*, Jun. 2010.
[34] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, 2nd ed. New York: Springer, Feb. 2009.
[35] P. Grünwald, *The Minimum Description Length Principle*. Cambridge, U.K.: MIT Press, Jun. 2007.
[36] Y. Shtarkov, "Universal sequential coding of single messages," *Probl. Inf. Transmiss.*, vol. 23, no. 3, pp. 3–17, Jul. 1987.
[37] H. Ishwaran and J. S. Rao, "Spike and slab variable selection: Frequentist and Bayesian strategies," *Ann. Statist.*, vol. 33, no. 2, pp. 730–773, 2005.
[38] T. M. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. 19, no. 1, pp. 73–77, 1973.
[39] G. Motta, E. Ordentlich, I. Ramirez, G. Seroussi, and M. Weinberger, "The iDUDE framework for grayscale image denoising," *IEEE Trans. Image Process.*, vol. 20, no. 1, pp. 1–21, Jan. 2011.
[40] E. Lam and J. Goodman, "A mathematical analysis of the DCT coefficient distributions for images," *IEEE Trans. Image Process.*, vol. 9, no. 10, pp. 1661–1666, 2000.
[41] R. E. Krichevsky and V. K. Trofimov, "The performance of universal encoding," *IEEE Trans. Inf. Theory*, no. 2, pp. 199–207, 1981.
[42] M. Zhou, H. Yang, G. Sapiro, D. Dunson, and L. Carin, "Dependent hierarchical beta process for image interpolation and denoising," in *Proc. AISTATS*, 2011.
[43] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Scientif. Comput.*, vol. 20, no. 1, pp. 33–61, 1998.
[44] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3d transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.
[45] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. ICCV*, Oct. 2009, pp. 2272–2279.
[46] E. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *J. ACM*, vol. 58, no. 3, May 2011.
[47] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization," in *Proc. Adv. NIPS*, Dec. 2009.
[48] Z. Lin, M. Chen, and Y. Ma, "The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices," [Online]. Available: http://arxiv.org/abs/1009.5055
[49] J. Rissanen, "A universal prior for integers and estimation by minimum description length," *Ann. Statist.*, vol. 11, no. 2, pp. 416–431, 1983.
[50] L. Li, W. Huang, I. Gu, and Q. Tian, "Statistical modeling of complex backgrounds for foreground object detection," *IEEE Trans. Image Process.*, vol. 13, no. 11, pp. 1459–1472, 2004.

**Ignacio Ramírez** (S'06) received the E.E. and the M.Sc. degrees in electrical engineering in 2002 and 2007, respectively, from the Universidad de la Repblica, Uruguay (UdelaR), and the Ph.D. degree in scientific computation in 2012 from the University of Minnesota (UofM), Minneapolis.

He was a Research Assistant at the UofM from 2008 to 2012. He also held temporary research positions at the UofM in 2007 and Hewlett-Packard Laboratories, Palo Alto, CA, in 2004. He holds an Assistantship with the Department of Electrical Engineering at UdelaR since 1999. His main research interests are applied information theory statistical signal processing and machine learning, with focus in multimedia data processing. His current research focuses in automatic model selection for sparse linear models.

**Guillermo Sapiro** was born in Montevideo, Uruguay, on April 3, 1966. He received the B.Sc. (*summa cum laude*), M.Sc., and Ph.D. degrees from the Department of Electrical Engineering, Technion, Israel Institute of Technology, in 1989, 1991, and 1993, respectively.

After postdoctoral research at the Massachusetts Institute of Technology (MIT), Cambridge , he became Member of Technical Staff at the research facilities of HP Labs, Palo Alto, CA. He is currently with the Department of Electrical and Computer Engineering, University of Minnesota, where he holds the position of Distinguished McKnight University Professor and Vincentine Hermes-Luh Chair in Electrical and Computer Engineering. He works on differential geometry and geometric partial differential equations, both in theory and applications in computer vision, computer graphics, medical imaging, and image analysis. He has authored and coauthored numerous papers in this area and has written a book published by Cambridge University Press in January 2001.

Dr. Sapiro recently co-edited a special issue of the IEEE TRANSACTIONS ON IMAGE PROCESSING in this topic and a second one in the *Journal of Visual Communication and Image Representation*. He was awarded the Gutwirth Scholarship for Special Excellence in Graduate Studies in 1991, the Ollendorff Fellowship for Excellence in Vision and Image Understanding Work in 1992, the Rothschild Fellowship for Post-Doctoral Studies in 1993, the Office of Naval Research Young Investigator Award in 1998, the Presidential Early Career Awards for Scientist and Engineers (PECASE) in 1998, the National Science Foundation Career Award in 1999, and the National Security Science and Engineering Faculty Fellowship in 2010. He is a member of SIAM. He is the founding Editor-in-Chief of the SIAM *Journal on Imaging Sciences*.