On MAP and MMSE Estimators for the Co-sparse Analysis Model[☆]

Javier S. Turek, Irad Yavneh, Michael Elad

Department of Computer Science, Technion, 3200003, Haifa, Israel

Abstract

The sparse synthesis model for signals has become very popular in the last decade, leading to improved performance in many signal processing applications. This model assumes that a signal may be described as a linear combination of few columns (*atoms*) of a given synthesis matrix (dictionary). The Co-Sparse Analysis model is a recently introduced counterpart, whereby signals are assumed to be orthogonal to many rows of a given analysis dictionary. These rows are called the co-support.

The Analysis model has already led to a series of contributions that address the pursuit problem: identifying the co-support of a corrupted signal in order to restore it. While all the existing work adopts a deterministic point of view towards the design of such pursuit algorithms, this paper introduces a Bayesian estimation point of view, starting with a random generative model for the co-sparse analysis signals. This is followed by a derivation of Oracle, Minimum-Mean-Squared-Error (MMSE), and Maximum-A'posteriori-Probability (MAP) based estimators. We present a comparison between the deterministic formulations and these estimators, drawing some connections between the two. We develop practical approximations to the MAP and MMSE estimators, and demonstrate the proposed reconstruction algorithms in several synthetic and real image experiments, showing their potential and applicability.

Keywords: Analysis Model, MAP, MMSE, Bayesian Estimation, Co-sparse Signal Model

[☆]The research leading to these results has received funding from the European Research Council under European Union's Seventh Framework Programme, ERC Grant agreement no. 320649.

Email addresses: javiert@cs.technion.ac.il (Javier S. Turek), irad@cs.technion.ac.il (Irad Yavneh), elad@cs.technion.ac.il (Michael Elad)

1. Introduction

In many signal and image processing tasks, such as denoising, deblurring, or inpainting problems, the main goal is to reconstruct a signal from a noisy degraded realization. In this paper we shall assume that an observation $\mathbf{y} \in \mathbb{R}^k$ is obtained by sensing a destination signal $\mathbf{x} \in \mathbb{R}^d$ using an observation matrix $\mathbf{M} \in \mathbb{R}^{k \times d}$ after it has been contaminated by additive noise $\mathbf{n} \in \mathbb{R}^k$, namely, $\mathbf{y} = \mathbf{M}\mathbf{x} + \mathbf{n}$. As this problem is typically ill-posed, the measurements alone do not suffice for recovering $\mathbf{x}$. The remedy is the introduction of a-priori knowledge about the class of signals to which $\mathbf{x}$ belongs, forcing some sort of regularity on the unknown signal. During the last decade, researchers in the fields of signal and image processing have become increasingly interested in sparse signal modelling as prior knowledge [1, 2, 3, 4].

Sparse modelling of signals assumes that a signal $\mathbf{x}$ can be written as a linear combination of columns (atoms) from a matrix (dictionary) $\mathbf{D} \in \mathbb{R}^{d \times m}$ ($m \geq d$) with coefficients from a sparse vector $\alpha \in \mathbb{R}^m$, i.e., $\mathbf{x} = \mathbf{D}\alpha$, and it is said that α is the sparse representation of $\mathbf{x}$ over the dictionary $\mathbf{D}$. The representation vector α has k non-zero elements, $\|\alpha\|_0 = k$, which is typically much smaller than the signal's dimension, d. This sparsity-inspired modelling is also called the Synthesis model, because it expresses how a signal $\mathbf{x}$ can be synthesized from its representation. This model has led to improved performance for many signal processing applications (see [5, 6, 7, 8, 9, 10] for representative works).

Recently, a counterpart to the above model was introduced [11, 12, 13, 14]: the Co-Sparse Analysis model. In this new approach, the signal $\mathbf{x}$ is defined by a known analysis dictionary $\mathbf{\Omega} \in \mathbb{R}^{p \times d}$, which can possibly be redundant, $p \geq d$. This model assumes that a signal satisfies $\|\mathbf{\Omega}\mathbf{x}\|_0 = p - \ell$, that is, the signal $\mathbf{x}$ is expected to be orthogonal to ℓ rows of $\mathbf{\Omega}$, and therefore to the subspace spanned by these ℓ rows. The subset Λ of ℓ orthogonal rows of $\mathbf{\Omega}$ is defined as the co-support. In other words, we say that $\mathbf{x}$ is ℓ-co-sparse, implying that its co-support Λ has ℓ elements and $\mathbf{\Omega}_\Lambda \mathbf{x} = \mathbf{0}$.

Suppose that a signal $\mathbf{x}$ is ℓ-co-sparse with respect to $\mathbf{\Omega}$. How can it be recovered from its observation $\mathbf{y}$? The following optimization task was proposed [11, 12, 13] for recovering the signal[1]:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{M}\mathbf{x}\|_2^2 + \lambda_2 \|\mathbf{x}\|_2^2 + \lambda_0 \|\mathbf{\Omega}\mathbf{x}\|_0 . \tag{1}$$

This optimization problem seems to be hard [13], because the ℓ_0 quasi-norm appears to require a sweep over all possible combinations of rows to be selected as the co-support. Methods like the Greedy Analysis Pursuit noise (GAPn) [16, 12], the Backward Greedy (BG) [17], and iterative projections algorithms [18, 19, 20] are employed to approximate a solution to the task in (1). Also, ℓ_1 based approximation can be considered, similar to the way it has been practiced for the synthesis sparse model [21, 22, 11, 16].

[1] Usually, it is assumed $\lambda_2 = 0$. The formulation posed in (1), with non-zero λ_2, is similar to the elastic net pursuit [15].

As an alternative to the deterministic approach described above, by assigning to the signal a prior probability density function, p.d.f., that is consistent with the Analysis model, one can derive Bayesian estimators for reconstructing $\mathbf{x}$. In the present work we introduce such an approach, and derive the related estimators. One estimator is the Minimum-Mean-Squared-Error (MMSE) and another option is the Maximum A-posteriori Probability (MAP). For MAP we distinguish between two alternatives - one that targets a co-support size, and another that considers the subspace dimension the signal should belong to. We refer to these two as MAPC (MAP-Co-support) and MAPS (MAP-Subspace), and we study the delicate differences between them. Note that all these estimators (MAP and MMSE) are hard to compute, with complexity that is exponential in p.

The estimators give rise to a series of questions: (i) When are these Bayesian estimators equivalent to the deterministic task in (1)? When they are not, which one should be preferred? (ii) How can we approximate the proposed estimators? In this work we aim to provide some answers to these questions. We give sufficient conditions for which the deterministic task becomes equivalent to the MAPC and MAPS estimators. Additionally, we present an efficient approximation algorithm to each estimator for reconstructing the signal $\hat{\mathbf{x}}$. We approximate the MMSE estimate by a Gibbs sampler [23], and the MAPC and MAPS by using a greedy pursuit approach [16, 17, 4]. In particular, the MAPS optimization task seems more difficult to approximate, and we suggest a way to overcome this obstacle. We compare the performance and computational aspects of these different methods and further discuss each option's pros and cons. Finally, we demonstrate the potential of these algorithms in a series of experiments involving both synthetic signals and real images.

The paper is organized as follows. In the next section, we briefly discuss the potential of the Co-Sparse Analysis model. In Section 3 the signal generation model is introduced. In Section 4 we derive the various estimators for $\hat{\mathbf{x}}$. Next, we expose a relation between MAP-based estimators and the deterministic problem in Section 5. Section 6 presents approximation algorithms and discusses their performance. In Section 7, we demonstrate the algorithms in synthetic and real-world signals. We conclude the work in Section 8.

2. Analysis vs. Synthesis Models

During the last decade, the Synthesis Sparse Representation model established itself and led to state-of-the-art results in many areas, and it seems that its younger counterpart, the Co-Sparse Analysis model, lags behind. One may argue why the Co-Sparse Analysis model should be chosen over the synthesis one? Indeed, the Co-Sparse Analysis model is still in development and it is not clear yet whether it is better or worse compared to the Synthesis. However, the two modeling approaches are known to be substantially different, and thus the Co-Sparse Analysis model may be found effective for specific data sources or some applications. We describe next some interesting work that has demonstrated the potential that exists within the Co-Sparse Analysis model.

In the classic image processing field, a recent work of Hawe et.al. [24, 25] has shown competitive results to that of K-SVD [26], BM3D [6], and Fields-of-Experts [27] for image denoising, inpainting, and super-resolution by learning an analysis dictionary and using it within the recovery process. In a different work, Portilla [28] showed highly competitive results for the deblurring problem, using ℓ_0 analysis with tight frame dictionaries. In [29], the authors compared results of synthesis and analysis methods using overcomplete wavelet dictionaries in 1D signals, obtaining better results in the analysis case. Additionally, the Analysis model has been used in other applications with promising results, for blind compressed sensing [30], MRI [31], color image super-resolution [32], and Poisson-Gaussian noise restoration [33].

Furthermore, another alternative and interesting recent approach is the integration of both models, using analysis and synthesis together. For example, the work by Danielyan et.al. [34] presented such an integrated model with patch-based adaptive frames as analysis and synthesis dictionaries, leading to state-of-the-art deblurring results. A follow-up paper by Ram et. al. [35] improved these deblurring results by modifying the frame construction, but still relying heavily on this merger of analysis and synthesis models. Shen et.al. [36, 37, 38] developed a "balanced" approach with analysis and synthesis frames integrated, and showed improved results for a variety of inverse problems in imaging. In particular, they observed that some visual artifacts appear in the synthesis-based approach while in the analysis-based and their suggested balanced approach the appearances of such artifacts are substantially reduced.

The Co-Sparse Analysis model is also found to be highly effective in recognition tasks, and perhaps this is the field that has benefitted the most from this model so far. The extensive work on deep-learning constructs auto-encoders that are used to extract sparsifying features, and this is done using the Analysis model. We refer the reader to the work in [39, 40, 41, 42] for further reading.

In all these cases, the Co-Sparse Analysis model is found to be of value. In this work our objective is not to convince the readers that the Analysis model is better than its Synthesis counterpart, but rather put forward algorithmic tools that would be useful for solving the elementary pursuit tasks that accompany practically any use of this model.

As a final remark, we should add that when it comes to the computational complexity of the pursuit task, if greedy algorithms are used, in the way we have practiced in this paper, then the synthesis approach has an advantage over the Analysis model. Synthesis greedy pursuit algorithms have a time complexity that is proportional to the number of non-zeros in the sparse representation, which is typically very small. On the other hand, analysis algorithms, such as MAPC and MAPS, as presented here after, are dependent on the co-rank, which is close to the signal dimension. In order to overcome this runtime gap, analysis methods may work in an opposite way, i.e., by adding non-zeros to the co-sparse vector $\mathbf{\Omega x}$ such as GAP and GAPn [16, 12] do for non-noisy signals. Another option is a relaxation-based method that computes the desired signal non-greedily. We consider these as important future directions of research.

3. An Analysis Signal Generation Model

Our goal is to obtain practical Bayesian estimation methods and this requires us to define a signal generation model that describes how a co-sparse signal $\mathbf{x} \in \mathbb{R}^d$ is created, and how a measurement vector $\mathbf{y} \in \mathbb{R}^k$ is related to it. The model introduced in this section is motivated by models posed for the synthesis model [43, 44, 45, 46, 47], adopting a probabilistic description.

Suppose that we want to draw an ℓ-co-sparse signal $\mathbf{x}$ under a given dictionary $\mathbf{\Omega} \in \mathbb{R}^{p \times d}$. This implies that there is a subset Λ of rows of $\mathbf{\Omega}$, the co-support, such that

$$\mathbf{\Omega}_\Lambda \mathbf{x} = \mathbf{0}. \tag{2}$$

This equation defines a subspace spanned by the rows of $\mathbf{\Omega}_\Lambda$ to which $\mathbf{x}$ is known to be orthogonal. We define a projection matrix[2] $\mathbf{Q}_\Lambda \in \mathbb{R}^{d \times d}$ that projects any vector into the subspace orthogonal to the one spanned by the rows of $\mathbf{\Omega}_\Lambda$,

$$\mathbf{Q}_\Lambda = \mathbf{I} - \mathbf{U}_\Lambda \mathbf{U}_\Lambda^T, \tag{3}$$

where $\mathbf{U}_\Lambda \in \mathbb{R}^{d \times \dim(\mathbf{\Omega}_\Lambda)}$ is obtained by applying an orthogonalization to the rows of $\mathbf{\Omega}_\Lambda$. It is easy to see that any $\mathbf{x}$ that obeys (2) satisfies $\mathbf{x} = \mathbf{Q}_\Lambda \mathbf{x}$.

Now we describe how to draw signals according to the Analysis model. We define two steps: (a) Drawing a co-support Λ, and (b) Drawing a signal $\mathbf{x}$ and its measurement $\mathbf{y}$ using Λ. We assign to each row in $\mathbf{\Omega}$ a Bernoulli distribution with probability q for a row being in the co-support. Hence, the co-support Λ is drawn by tossing p coins, each one with probability q. The a-priori probability of the co-support Λ is then given by

$$P(\Lambda) = q^{|\Lambda|} \cdot (1-q)^{p-|\Lambda|} = \left(\frac{q}{1-q}\right)^{|\Lambda|} \cdot (1-q)^p. \tag{4}$$

Note that the co-support Λ can be of any size, while its expected size is pq. Once the co-support Λ is drawn, the matrix $\mathbf{Q}_\Lambda$ is obtained from Equation (3).

The next step is to draw the signal itself using the co-support Λ. We draw a vector $\mathbf{x_0} \in \mathbb{R}^d$ from a multivariate Gaussian distribution, i.e., $\mathcal{N}\left(\mathbf{0}, \sigma_x^2 \mathbf{I}\right)$. Then, $\mathbf{x_0}$ is projected by $\mathbf{Q}_\Lambda$ so that the outcome satisfies the constraint in Equation (2). The resulting projected vector $\mathbf{x} = \mathbf{Q}_\Lambda \mathbf{x_0}$ is our clean co-sparse signal. Finally, a noisy measurement $\mathbf{y}$ of the signal $\mathbf{x}$ is obtained by applying the observation operator $\mathbf{M}$ and adding a white Gaussian i.i.d. noise vector,

$$\mathbf{y} = \mathbf{M}\mathbf{x} + \mathbf{n} = \mathbf{M}\mathbf{Q}_\Lambda \mathbf{x_0} + \mathbf{n}, \tag{5}$$

where $\mathbf{n} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \mathbf{I}\right)$ is the additive noise.

[2] Alternatively, a projection matrix $\mathbf{Q}_\Lambda$ can be computed using the pseudo-inverse $\mathbf{\Omega}_\Lambda^\dagger$. However, linear dependencies in $\mathbf{\Omega}_\Lambda$ should be treated before computing the projection matrix in this manner.

Given the co-support Λ, the signal $\mathbf{y}$ is obtained by linear operations on Gaussian vectors, and therefore we have

$$\mathbf{y}|\Lambda \sim \mathcal{N}\left(\mathbf{0}, \sigma_x^2 \mathbf{C}_\Lambda\right), \tag{6}$$

where the matrix $\mathbf{C}_\Lambda$ is defined by

$$\mathbf{C}_\Lambda = \mathbf{M}\mathbf{Q}_\Lambda\mathbf{M}^T + \frac{\sigma^2}{\sigma_x^2}\mathbf{I}. \tag{7}$$

Similarly, the posterior distribution of $\mathbf{x}$ given $\mathbf{y}$ and the co-support [48, 49] is given by

$$\mathbf{x}|\mathbf{y}, \Lambda \sim \mathcal{N}\left(\mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{C}_\Lambda^{-1}\mathbf{y}, \sigma_x^2\mathbf{Q}_\Lambda - \sigma^2\mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{C}_\Lambda^{-1}\mathbf{M}\mathbf{Q}_\Lambda\right). \tag{8}$$

These definitions are useful for deriving the estimators in the following sections.

4. Bayesian Estimators

In the previous section we described the random signal generation model, and now we develop several estimators to recover $\mathbf{x}$: the Oracle estimator that knows the co-support, the Minimum Mean Squared Error (MMSE) estimator, the Maximum A-Posteriori Co-support (MAPC) estimator, and a Maximum A-Posteriori Subspace (MAPS) estimator. The oracle estimator is of theoretical value only, but is important in the path we take, due to two reasons: First, it gives us a lower bound on the attainable estimation error in practical methods. In addition, as we shall see, its expressions will serve for the derivation of the other estimators.

4.1. Oracle Estimator

The Oracle estimator assumes that partial, but helpful, additional information regarding $\mathbf{x}$ is given. For our model, this information is the co-support Λ. Assuming that Λ is given, the projection matrix $\mathbf{Q}_\Lambda$ is known, and as we have seen already in Section 2, the posterior p.d.f. of $\mathbf{x}|\mathbf{y}, \Lambda$, is a Gaussian multivariate distribution, as given in Equation (8). In this case, estimating the original vector $\mathbf{x}$ can be done with the Minimum Mean Squared Error (MMSE) estimator or the Maximum A-posteriori Probability (MAP) estimator, obtaining with both the same estimate:

$$\hat{\mathbf{x}}_\Lambda^O = E\left\{\mathbf{x}|\mathbf{y}, \Lambda\right\} = \mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{C}_\Lambda^{-1}\mathbf{y} = \left(\frac{\sigma^2}{\sigma_x^2}\mathbf{I} + \mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{M}\mathbf{Q}_\Lambda\right)^{-1}\mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{y}, \tag{9}$$

where the last equality is obtained by applying the matrix inversion lemma on $\mathbf{C}_\Lambda$ (see Appendix A).

4.2. Minimum-Mean-Squared-Error (MMSE) Estimator

The MMSE estimate minimizes the mean-squared-error (MSE) and it is given by the conditional expectation, $E\{\mathbf{x}|\mathbf{y}\}$. Marginalizing the conditional expectation over all possible co-supports Λ we have

$$\begin{aligned}\hat{\mathbf{x}}^{MMSE} &= E\{\mathbf{x}|\mathbf{y}\} \\ &= \frac{1}{t}\sum_{\Lambda\in\Gamma} E\{\mathbf{x}|\mathbf{y},\Lambda\}\, P(\Lambda|\mathbf{y}) \\ &= \frac{1}{t}\sum_{\Lambda\in\Gamma} P(\Lambda|\mathbf{y})\,\hat{\mathbf{x}}_\Lambda^O, \end{aligned} \tag{10}$$

where Γ is the set of all possible co-supports, $t = \sum_{\Lambda\in\Gamma} P(\Lambda|\mathbf{y})$ is a normalization constant, and $\hat{\mathbf{x}}_\Lambda^O$ is the oracle estimate in Equation (9) using a specific co-support Λ. We now derive the posterior p.d.f. of the support $P(\Lambda|\mathbf{y})$. First, we apply Bayes's rule to obtain

$$P(\Lambda|\mathbf{y}) = \frac{P(\mathbf{y}|\Lambda)\,P(\Lambda)}{P(\mathbf{y})} \propto P(\mathbf{y}|\Lambda)\,P(\Lambda)\,. \tag{11}$$

Let us recall the term $P(\mathbf{y}|\Lambda)$ from the distribution in Equation (6),

$$P(\mathbf{y}|\Lambda) = \frac{1}{\sqrt{(2\pi\sigma_x^2)^k \det(\mathbf{C}_\Lambda)}} \exp\left\{-\frac{1}{2\sigma_x^2}\mathbf{y}^T\mathbf{C}_\Lambda^{-1}\mathbf{y}\right\}. \tag{12}$$

Plugging $P(\Lambda)$ from Equation (4) and Equation (12) back into Equation (11) we obtain

$$P(\Lambda|\mathbf{y}) \propto \frac{1}{\sqrt{\det(\mathbf{C}_\Lambda)}} \exp\left\{-\frac{1}{2\sigma_x^2}\mathbf{y}^T\mathbf{C}_\Lambda^{-1}\mathbf{y}\right\}\left(\frac{q}{1-q}\right)^{|\Lambda|}. \tag{13}$$

Next, let us substitute Equation (13) back into the $\hat{\mathbf{x}}^{MMSE}$ estimate in Equation (10), obtaining

$$\hat{\mathbf{x}}^{MMSE} = \frac{1}{t}\sum_{\Lambda\in\Gamma} \frac{1}{\sqrt{\det(\mathbf{C}_\Lambda)}} e^{\left\{-\frac{1}{2\sigma_x^2}\mathbf{y}^T\mathbf{C}_\Lambda^{-1}\mathbf{y}\right\}}\left(\frac{q}{1-q}\right)^{|\Lambda|}\hat{\mathbf{x}}_\Lambda^O. \tag{14}$$

This formula implies an averaging over all possible co-supports to compute the MMSE estimate. Note that this averaging suggests that the MMSE estimate, $\mathbf{\Omega}\hat{\mathbf{x}}^{MMSE}$ is in fact not co-sparse. This phenomenon appears also in the MMSE estimator for the synthesis model in [43, 45, 44].

To compute the MMSE estimation, a sweep over all possible co-supports Λ in Γ is needed. This set contains 2^p different co-supports, implying that this computation is infeasible in general.

Note that in [43] we developed a Bayesian estimation solution for denoising of signals emerging from the synthesis model with a *unitary* dictionary. This

case (unitary synthesis model) is in fact completely equivalent to an Analysis model with the inverse of the above dictionary as the analysis dictionary. In this special case, [43] shows that the exact MMSE is in fact computable, and there is a closed-form shrinkage solution to the estimation.

The results in [43] are not applicable in this paper because of two important reasons: (i) we consider general and redundant analysis dictionaries and we are no longer restricted to unitary ones; and (ii) in this work we treat general inverse problems, where denoising is merely a simple special case.

4.3. Maximum A-posteriori Probability for Co-support (MAPC) Estimator

The general and well known MAP estimator finds an estimate $\hat{\mathbf{x}}$ that maximizes the posterior probability, $P(\mathbf{x}|\mathbf{y})$. Because our model mixes discrete probabilities q with continuous ones $P(\mathbf{y}|\Lambda)$, the MAPC should be carefully formulated. For that reason, we must find first the MAP co-support $\hat{\mathbf{\Lambda}}^{MAPC}$ by maximizing the co-support posterior $P(\Lambda|\mathbf{y})$,

$$\hat{\mathbf{\Lambda}}^{MAPC} = \arg\max_{\Lambda\in\Gamma} P(\Lambda|\mathbf{y}), \tag{15}$$

and only then compute the associated MAPC estimate, $\hat{\mathbf{x}}^{MAPC}$ as the oracle on the found co-support.

Plugging Equation (13) into Equation (15), taking the log and inverting the sign of the penalty function while minimizing over all possible co-supports, we obtain the following optimization task for the MAPC estimator

$$\hat{\mathbf{\Lambda}}^{MAPC} = \arg\min_{\Lambda\in\Gamma} \frac{1}{2\sigma_x^2}\mathbf{y}^T\mathbf{C}_\Lambda^{-1}\mathbf{y} + \frac{1}{2}\ln\left[\det\left(\mathbf{C}_\Lambda\right)\right] - |\Lambda|\ln\left(\frac{q}{1-q}\right). \tag{16}$$

After the co-support $\hat{\mathbf{\Lambda}}^{MAPC}$ is computed, the MAPC estimate $\hat{\mathbf{x}}^{MAPC}$ is obtained by substituting this co-support into the Oracle formula in Equation (9),

$$\hat{\mathbf{x}}^{MAPC} = \mathbf{Q}_{\hat{\Lambda}^{MAPC}}\mathbf{M}^T\mathbf{C}^{-1}_{\hat{\Lambda}^{MAPC}}\mathbf{y}. \tag{17}$$

Observe that the minimization task in Equation (16) sweeps over all the possible co-supports Λ in Γ to compute $\hat{\mathbf{\Lambda}}^{MAPC}$, suggesting again that computing this estimation is as hard as the MMSE. Just as for the MMSE, the work in [43] derived a close-form shrinkage solution for the MAP estimation. However, this is relevant only for unitary dictionaries, and thus these results are not applicable here.

4.4. Maximum A-posteriori Probability for Subspace (MAPS) Estimator

Section 3 describes how a signal $\mathbf{x}$ is projected onto the subspace spanned by the rows of $\mathbf{\Omega}_\Lambda$, where Λ is the co-support drawn for that signal. When the dictionary $\mathbf{\Omega}$ is redundant, i.e., $p > d$, there are linear dependencies between rows of $\mathbf{\Omega}$. Consequently, different co-supports may span the same subspace. Hence, the probability of a signal to be drawn with that subspace increases. Due to this fact, we suggest an alternative MAP – Maximum A-posteriori Probability

for Subspace (MAPS) estimator, that aims to maximize the subspace posterior p.d.f.. Like MAPC, this estimator is MAP-based and it should be carefully derived, thus we find first the MAP subspace $\hat{\mathbf{\Psi}}^{MAPS}$, and then compute the MAPS estimate $\hat{\mathbf{x}}^{MAPS}$ using the Oracle formula in Equation (9) with the subspace $\hat{\mathbf{\Psi}}^{MAPS}$.

Let us define $\Lambda_{\mathbf{\Psi}}$ as the biggest co-support such that the rows of $\mathbf{\Omega}_{\Lambda_{\mathbf{\Psi}}}$ span the subspace $\mathbf{\Psi}$. Let $P(\mathbf{\Psi})$ be the probability of this subspace $\mathbf{\Psi}$, defined as the sum of all the probabilities of the co-supports that span it,

$$P(\mathbf{\Psi}) = \sum_{\substack{\Phi \subseteq \Lambda_{\mathbf{\Psi}}, \\ \text{span}\,(\mathbf{\Omega}_{\Phi}) = \mathbf{\Psi}}} P(\Phi)\,. \tag{18}$$

Using Bayes's rule and Equation (18), the MAPS estimator is defined as follows

$$\begin{aligned} \hat{\mathbf{\Psi}}^{MAPS} &= \arg\max_{\Psi} P(\mathbf{\Psi}|\mathbf{y}) \\ &= \arg\max_{\mathbf{\Psi}} P(\mathbf{y}|\mathbf{\Psi})\, P(\mathbf{\Psi}) \\ &= \arg\max_{\mathbf{\Psi}} P(\mathbf{y}|\mathbf{\Psi}) \sum_{\substack{\Phi \subseteq \Lambda_{\mathbf{\Psi}}, \\ \text{span}\,(\mathbf{\Omega}_{\Phi}) = \mathbf{\Psi}}} P(\Phi)\,, \end{aligned} \tag{19}$$

where $P(\mathbf{y}|\mathbf{\Psi})$ is the likelihood of a signal given a subspace, computed using Equation (12) with any co-support that spans $\mathbf{\Psi}$, (e.g., $\Lambda_{\mathbf{\Psi}}$). Applying similar algebraic steps as with MAPC, we can rewrite Equation (19) as follows,

$$\begin{aligned} \hat{\mathbf{\Psi}}^{MAPS} = \arg\min_{\mathbf{\Psi}} \quad & \frac{1}{2\sigma_x^2}\mathbf{y}^T\mathbf{C}_{\mathbf{\Psi}}^{-1}\mathbf{y} + \frac{1}{2}\ln\left[\det\left(\mathbf{C}_{\mathbf{\Psi}}\right)\right] \\ & - \ln\left(\sum_{\substack{\Phi \subseteq \Lambda_{\mathbf{\Psi}} \\ \text{span}\,(\mathbf{\Omega}_{\Phi}) = \mathbf{\Psi}}} \left(\frac{q}{1-q}\right)^{|\Phi|}\right), \end{aligned} \tag{20}$$

where $\mathbf{C}_{\mathbf{\Psi}}$ is defined in Equation (7) but computed with one of the co-supports that spans $\mathbf{\Psi}$. After computing the subspace $\hat{\mathbf{\Psi}}^{MAPS}$, the MAPS estimate $\hat{\mathbf{x}}^{MAPS}$ is obtained using the Oracle formula (9),

$$\hat{\mathbf{x}}^{MAPS} = \mathbf{Q}_{\hat{\mathbf{\Psi}}^{MAPS}}\mathbf{M}^T\mathbf{C}^{-1}_{\hat{\mathbf{\Psi}}^{MAPS}}\mathbf{y}. \tag{21}$$

Similarly to the previous estimators, the MAPS estimator sweeps over all possible subspaces (and their co-supports), requiring a sweep over all 2^p co-supports. Therefore, the complexity of this estimator is as hard as the previous MAPC and MMSE estimators.

The difference between the optimization task for MAPC estimation in Equation (16) and that of MAPS in Equation (20) is in the last term, which is given

by the sum of the co-support probabilities $P(\Phi)$. The signal generation procedure presented in Section 3 describes a model where the same subspace can be drawn for different co-supports, increasing the probability that such a subspace be chosen for the signal. In fact, some subspaces that are spanned by a bigger number of combinations of rows from $\mathbf{\Omega}$, become more probable in comparison to their probability in MAPC estimation. This is the reason why this new estimate formulation is of interest.

The change of view from co-support to subspace as in MAPS, may be done in the MMSE estimator as well. The MMSE estimator can be derived using the subspace posterior p.d.f. to obtain a summation of all the possible subspaces, instead of co-supports as in Equation (14). However, there is no numerical difference between these two possible derivations. The probability of a subspace is equal to the sum of the probabilities of the co-supports which span that subspace, and the oracle estimate is the same for all of them. Therefore, there is no point in studying a MMSE formula with a sweep over all the subspaces.

5. MAP Versus Deterministic Pursuit

In this section we show a relation between the deterministic problem in (1) and the MAP-based estimators described in the previous section. This connection refers to the equivalence between the deterministic problem and the optimization problems solved to compute the MAP-based estimators (for both MAPC and MAPS) when the dictionary has specific characteristics. This fact allows us to define closed-form formulas based on the model parameters for the regularizers λ_0 and λ_2 in the deterministic problem.

5.1. Preliminaries

Let us start by proving the following Lemmas that are useful in the proof of the main theorem below.

Lemma 1. *Given a dictionary $\mathbf{\Omega}$, a co-support Λ, and a measurement $\mathbf{y}$, the following two optimization tasks:*

$$(A) \qquad \mathbf{x}_A = \arg\min_{\mathbf{x}} \|\mathbf{y}-\mathbf{M}\mathbf{x}\|_2^2 + \lambda_2 \|\mathbf{x}\|_2^2 \quad \text{s.t.} \quad \mathbf{\Omega}_\Lambda \mathbf{x} = 0,$$

and

$$(B) \qquad \mathbf{x}_B = \arg\min_{\mathbf{x}} \|\mathbf{y}-\mathbf{M}\mathbf{Q}_\Lambda\mathbf{x}\|_2^2 + \lambda_2 \|\mathbf{x}\|_2^2 ,$$

where $\mathbf{Q}_\Lambda$ is given in Equation (3)*, and $\lambda_2 > 0$, have the same minimizer, i.e., $\mathbf{x}_A = \mathbf{x}_B$.*

Proof: Let $\mathbf{x}$ be a feasible solution of problem (A), thus satisfying the constraint $\mathbf{\Omega}_\Lambda \mathbf{x} = 0$. Then, $\mathbf{Q}_\Lambda \mathbf{x} = \mathbf{x}$ and we have that $\|\mathbf{y}-\mathbf{M}\mathbf{x}\|_2^2 + \lambda_2 \|\mathbf{x}\|_2^2 = \|\mathbf{y}-\mathbf{M}\mathbf{Q}_\Lambda\mathbf{x}\|_2^2 + \lambda_2 \|\mathbf{x}\|_2^2$, which is the same penalty as in problem (B). In particular, the optimizer of (A), $\mathbf{x}_A$, is one feasible solution that satisfies the

constraint, hence $\|\mathbf{y}-\mathbf{M}\mathbf{Q}_\Lambda\mathbf{x}_B\|_2^2+\lambda_2\|\mathbf{x}_B\|_2^2 \le \|\mathbf{y}-\mathbf{M}\mathbf{Q}_\Lambda\mathbf{x}_A\|_2^2+\lambda_2\|\mathbf{x}_A\|_2^2 = \|\mathbf{y}-\mathbf{M}\mathbf{x}_A\|_2^2+\lambda_2\|\mathbf{x}_A\|_2^2$.

We now show that the minimizer of (B) satisfies the constraint $\mathbf{\Omega}_\Lambda\mathbf{x}=0$. Let us decompose $\mathbf{x}$ as $\mathbf{x}=\mathbf{Q}_\Lambda\mathbf{x}+\bar{\mathbf{x}}$, where the vectors $\mathbf{Q}_\Lambda\mathbf{x}$ and $\bar{\mathbf{x}}$ are orthogonal to each other, i.e., $\bar{\mathbf{x}}^T\mathbf{Q}_\Lambda\mathbf{x}=0$. In particular, $\mathbf{Q}_\Lambda\bar{\mathbf{x}}=0$. The penalty function in (B) is then given by

$$\begin{aligned}&\|\mathbf{y}-\mathbf{M}\mathbf{Q}_\Lambda\left(\mathbf{Q}_\Lambda\mathbf{x}+\bar{\mathbf{x}}\right)\|_2^2+\lambda_2\|\mathbf{Q}_\Lambda\mathbf{x}+\bar{\mathbf{x}}\|_2^2\\&=\|\mathbf{y}-\mathbf{M}\mathbf{Q}_\Lambda\mathbf{x}\|_2^2+\lambda_2\|\mathbf{Q}_\Lambda\mathbf{x}\|_2^2+\lambda_2\|\bar{\mathbf{x}}\|_2^2.\end{aligned}$$

Clearly, the term $\|\bar{\mathbf{x}}\|_2^2$ must be zero for $\mathbf{x}_B$, otherwise we can reduce the functional by subtracting $\bar{\mathbf{x}}$ from $\mathbf{x}_B$. Hence, the minimizer $\mathbf{x}_B$ satisfies the constraint $\mathbf{\Omega}_\Lambda\mathbf{x}_B=0$ in the optimization task (A), and it is part of the feasible domain. In particular, we obtain $\|\mathbf{y}-\mathbf{M}\mathbf{x}_A\|_2^2+\lambda_2\|\mathbf{x}_A\|_2^2 \le \|\mathbf{y}-\mathbf{M}\mathbf{x}_B\|_2^2+\lambda_2\|\mathbf{x}_B\|_2^2$.

Problems (A) and (B) are convex optimization problems with strictly convex penalty functions of quadratic form (where $\mathbf{Q}_\Lambda^T\mathbf{M}^T\mathbf{M}\mathbf{Q}_\Lambda+\lambda_2\mathbf{I}$ and $\mathbf{M}^T\mathbf{M}+\lambda_2\mathbf{I}$ are positive definite matrices). From this and the previous arguments, it follows that the minimizer of (A) and the minimizer of (B) are the same, i.e. $\mathbf{x}_A=\mathbf{x}_B$. □

The following Lemma connects between the two MAP-based estimators we introduced earlier - the MAPC and the MAPS. (1).

Lemma 2. *If the analysis dictionary* $\mathbf{\Omega}$ *is in "general position", i.e.* $\dim(\mathbf{\Omega}_\Lambda)=|\Lambda|$ *for any co-support* Λ *satisfying* $|\Lambda|\le d$*, then* $\hat{\mathbf{x}}^{MAPC}=\hat{\mathbf{x}}^{MAPS}$.

Proof: From the fact that there are no linear dependencies in $\mathbf{\Omega}$, every co-support Λ with $|\Lambda|\le d$, defines a subset of linearly independent rows $\mathbf{\Omega}_\Lambda$ that span a distinct subspace. Therefore, the last term from the MAPS penalty in Equation (20) reduces to only one co-support that spans that subspace. In such a case, the penalty of the MAPS optimization task reduces exactly to the MAPC optimization task proving that $\hat{\mathbf{\Lambda}}^{MAPC}=\hat{\mathbf{\Lambda}}^{MAPS}$ and also $\hat{\mathbf{x}}^{MAPC}=\hat{\mathbf{x}}^{MAPS}$. □

5.2. Main Result - Deterministic-Bayesian Equivalence

Now we turn to show the relation between the MAPC estimate and the deterministic problem posed in Equation (1):

Theorem 3. *Assuming that (i) the analysis dictionary* $\mathbf{\Omega}$ *is in general position, i.e.,* $|\Lambda|=\dim(\mathbf{\Omega}_\Lambda)$ *for any co-support* Λ *such that* $|\Lambda|\le d$*, (ii) the observation matrix* $\mathbf{M}$ *satisfies* $\mathbf{M}^T\mathbf{M}=c\mathbf{I}$ *(*$c>0$*), and (iii) the noisy signal* $\mathbf{y}$ *is contaminated with white Gaussian i.i.d. noise* $\mathcal{N}\left(\mathbf{0},\sigma^2\mathbf{I}\right)$*, then the MAPC estimate, as defined in Equations* (16) *and* (17)*, and the deterministic pursuit, as defined in Equation* (1)*, are equivalent, where* $\lambda_2=\frac{\sigma^2}{\sigma_x^2}$ *and* $\lambda_0=2\sigma^2\ln\left[\frac{q}{1-q}\sqrt{\frac{\sigma^2+c\sigma_x^2}{\sigma^2}}\right]$.

Proof: First, let us recall how the MAPC and the deterministic pursuit operate. The MAPC estimation is obtained by solving for the MAPC co-support

$$\hat{\mathbf{\Lambda}}^{MAPC} = \arg\min_{\Lambda\in\Gamma} \quad \frac{1}{2\sigma^2}\mathbf{y}^T\mathbf{y} - \frac{1}{2\sigma^2}\mathbf{y}^T\mathbf{M}\mathbf{Q}_\Lambda\left(\frac{\sigma^2}{\sigma_x^2}\mathbf{I} + \mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{M}\mathbf{Q}_\Lambda\right)^{-1}\mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{y} + \frac{1}{2}\ln\left[\det\left(\mathbf{C}_\Lambda\right)\right] - |\Lambda|\ln\left(\frac{q}{1-q}\right), \tag{22}$$

and computing the estimate $\hat{\mathbf{x}}^{MAPC} = \mathbf{Q}_{\hat{\Lambda}^{MAPC}}\mathbf{M}^T\mathbf{C}^{-1}_{\hat{\Lambda}^{MAPC}}\mathbf{y}$, whereas the deterministic pursuit is computed by solving

$$\hat{\mathbf{x}}^{DET} = \arg\min_{\mathbf{x}} \|\mathbf{y}-\mathbf{M}\mathbf{x}\|_2^2 + \lambda_2\|\mathbf{x}\|_2^2 + \lambda_0\|\mathbf{\Omega}\mathbf{x}\|_0. \tag{23}$$

Let Λ be a co-support with respect to $\mathbf{x}$, such that $\mathbf{\Omega}_\Lambda\mathbf{x} = \mathbf{0}$ and $|\Lambda| \le d$. By definition, we have that $\|\mathbf{\Omega}\mathbf{x}\|_0 = p - |\Lambda|$ for the associated $\mathbf{x}$. Therefore, by requiring $\mathbf{\Omega}_\Lambda\mathbf{x} = \mathbf{0}$, we rewrite the problem in (23) as a constrained minimization over both $\mathbf{x}$ and Λ as follows

$$\arg\min_{\Lambda} \quad \left[\arg\min_{\mathbf{x}} \|\mathbf{y}-\mathbf{M}\mathbf{x}\|_2^2 + \lambda_2\|\mathbf{x}\|_2^2 + \lambda_0\left(p-|\Lambda|\right)\right] \quad \text{s.t.} \quad \mathbf{\Omega}_\Lambda\mathbf{x} = 0. \tag{24}$$

Using Lemma 1 we rewrite this in unconstrained form:

$$\arg\min_{\Lambda}\left[\arg\min_{\mathbf{x}} \|\mathbf{y}-\mathbf{M}\mathbf{Q}_\Lambda\mathbf{x}\|_2^2 + \lambda_2\|\mathbf{x}\|_2^2 + \lambda_0\left(p-|\Lambda|\right)\right]. \tag{25}$$

Let us calculate the optimal solution $\hat{\mathbf{x}}$ of the inner minimization task (25), assuming that Λ is given. We differentiate the penalty function $J(\mathbf{x};\Lambda) = \|\mathbf{y}-\mathbf{M}\mathbf{Q}_\Lambda\mathbf{x}\|_2^2 + \lambda_2\|\mathbf{x}\|_2^2 + \lambda_0\left(p-|\Lambda|\right)$ and set it equal to $\mathbf{0}$:

$$\mathbf{0} = \frac{\partial J(\mathbf{x};\Lambda)}{\partial\mathbf{x}} = -2\mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{y} + 2\mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{M}\mathbf{Q}_\Lambda\mathbf{x} + 2\lambda_2\mathbf{x}.$$

Then, the optimal $\hat{\mathbf{x}}$ for a given co-support Λ is given by

$$\begin{aligned}\hat{\mathbf{x}} &= \left(\lambda_2\mathbf{I} + \mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{M}\mathbf{Q}_\Lambda\right)^{-1}\mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{y} \\ &= \mathbf{Q}_\Lambda\mathbf{M}^T\left(\lambda_2\mathbf{I} + \mathbf{M}^T\mathbf{Q}_\Lambda\mathbf{M}\right)^{-1}\mathbf{y},\end{aligned} \tag{26}$$

where the last step is obtained by applying the matrix inversion lemma (see Appendix A). Note the similarity between $\hat{\mathbf{x}}^{MAPC}$ in problem (22) and $\hat{\mathbf{x}}$ (up to the yet unknown co-support).

Plugging Equation (26) into the optimization task in (25), we obtain the penalty to be minimized with respect to the co-support

$$\begin{aligned}\arg\min_{\Lambda} J(\hat{\mathbf{x}};\Lambda) &= \arg\min_{\Lambda}\left\|\mathbf{y}-\mathbf{M}\mathbf{Q}_\Lambda\left(\lambda_2\mathbf{I}+\mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{M}\mathbf{Q}_\Lambda\right)^{-1}\mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{y}\right\|_2^2 \\ &\quad +\lambda_2\left\|\left(\lambda_2\mathbf{I}+\mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{M}\mathbf{Q}_\Lambda\right)^{-1}\mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{y}\right\|_2^2 + \lambda_0\left(p-|\Lambda|\right) \\ &= \arg\min_{\Lambda}\|\mathbf{y}\|_2^2 - \mathbf{y}^T\mathbf{M}\mathbf{Q}_\Lambda\left(\lambda_2\mathbf{I}+\mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{M}\mathbf{Q}_\Lambda\right)^{-1}\mathbf{Q}_\Lambda\mathbf{M}^T\mathbf{y} \\ &\quad +\lambda_0\left(p-|\Lambda|\right).\end{aligned} \tag{27}$$

Equation (27) above shows signs of similarity (up to a constant) with the penalty of the MAPC estimator in Equation (22). Next, we compare both penalty functions to find the values for the parameters λ_2 and λ_0 to make the problems identical. By comparing the solutions $\hat{\mathbf{x}}$ in Equation (26) and the MAPC estimate in Equation (17) we have

$$\lambda_2 = \frac{\sigma^2}{\sigma_x^2}. \tag{28}$$

Now, we compute the determinant of the matrix $\mathbf{C}_\Lambda$ for a given co-support Λ:

$$\begin{aligned}
\det\left(\mathbf{C}_{\mathbf{\Lambda}}\right) &= \det\left(\frac{\sigma^{\mathbf{2}}}{\sigma_{\mathbf{x}}^{\mathbf{2}}}\mathbf{I} + \mathbf{M}\mathbf{Q}_{\mathbf{\Lambda}}\mathbf{M}^{\mathbf{T}}\right) \\
&= \left(\frac{\sigma^2}{\sigma_x^2}\right)^k \det\left(\mathbf{I} + \frac{\sigma_{\mathbf{x}}^{\mathbf{2}}}{\sigma^{\mathbf{2}}}\mathbf{M}\mathbf{Q}_{\mathbf{\Lambda}}\mathbf{M}^{\mathbf{T}}\right) \\
&= \left(\frac{\sigma^2}{\sigma_x^2}\right)^k \det\left(\mathbf{I} + \frac{\sigma_{\mathbf{x}}^{\mathbf{2}}}{\sigma^{\mathbf{2}}}\mathbf{M}^{\mathbf{T}}\mathbf{M}\mathbf{Q}_{\mathbf{\Lambda}}\right) \\
&= \left(\frac{\sigma^2}{\sigma_x^2}\right)^k \det\left(\mathbf{I} + \mathbf{c}\frac{\sigma_{\mathbf{x}}^{\mathbf{2}}}{\sigma^{\mathbf{2}}}\mathbf{Q}_{\mathbf{\Lambda}}\right) \\
&= \left(\frac{\sigma^2}{\sigma_x^2}\right)^k \left(1 + c\frac{\sigma_x^2}{\sigma^2}\right)^{d-\dim(\mathbf{\Omega}_{\mathbf{\Lambda}})},
\end{aligned} \tag{29}$$

where in the second step a determinant identity is used (see Appendix A), and the assumption $\mathbf{M}^T\mathbf{M} = c\mathbf{I}$ is used in the third step. As there are no linear dependencies in $\mathbf{\Omega}_\Lambda$, $\dim\left(\mathbf{\Omega}_{\mathbf{\Lambda}}\right)$ can be substituted by the co-support size $|\Lambda|$ in Equation (29).

Finally, compare Equation (27) to the MAPC optimization task in (22) and substitute the determinant of $\mathbf{C}_\Lambda$ in Equation (29), while assuming $\dim\left(\mathbf{\Omega}_{\mathbf{\Lambda}}\right) = |\Lambda|$. We get

$$\lambda_0 = 2\sigma^2 \ln\left(\frac{q}{1-q}\sqrt{\frac{\sigma^2 + c\sigma_x^2}{\sigma^2}}\right), \tag{30}$$

which completes the proof.

□

This theorem yields a relation between the Bayesian and deterministic approaches. This relation is limited to observation matrices satisfying $\mathbf{M}^T\mathbf{M} = c\mathbf{I}$ and general positioned $\mathbf{\Omega}$. In more general scenarios, where linear dependencies in $\mathbf{\Omega}$ do exist, there are differences between these three methods (MAPC, MAPS, and the deterministic pursuit). These differences occur in the last two terms of Equations (22), (20), and (1) and correspond to the co-support size $|\Lambda|$ or the dimension $\dim\left(\mathbf{\Omega}_{\mathbf{\Lambda}}\right)$ of the subspace spanned by the rows of $\mathbf{\Omega}_\Lambda$.

5.3. Denoising

In the particular task of denoising, whereupon $\mathbf{M} = \mathbf{I}$, the observation matrix requirement in Theorem 3 is satisfied. Therefore, we use this problem to compare the estimators and the deterministic problem, for non-general positioned dictionaries. For denoising, the solution $\hat{\mathbf{x}}$ in Equation (26) is simplified to

$$\hat{\mathbf{x}} = \left(\lambda_2 \mathbf{I} + \mathbf{Q}_\Lambda\right)^{-1} \mathbf{Q}_\Lambda \mathbf{y}.$$

Obviously, the oracle estimator formula in (9) is the same. We can substitute $\mathbf{Q}_\Lambda$ using its singular value decomposition $\mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$, and rewrite the above formula as

$$\hat{\mathbf{x}} = \mathbf{U}\left(\lambda_2 \mathbf{I} + \mathbf{\Sigma}\right)^{-1} \mathbf{\Sigma}\mathbf{U}^T \mathbf{y}.$$

We note that the diagonal matrix $\mathbf{\Sigma}$ contains $d - \dim\left(\mathbf{\Omega}_{\mathbf{\Lambda}}\right)$ ones, and zeros otherwise. Thus, the diagonal matrix $\left(\lambda_2 \mathbf{I} + \mathbf{\Sigma}\right)^{-1} \mathbf{\Sigma}$ contains $d - \dim\left(\mathbf{\Omega}_{\mathbf{\Lambda}}\right)$ elements with value $\frac{1}{1+\lambda_2}$ and zeros elsewhere. As a result, this matrix functions as a hard thresholding operator, zeroing all the components in the subspace spanned by the rows of $\mathbf{\Omega}_\Lambda$, and shrinking the part of the signal that resides in the orthogonal subspace by a factor $\frac{1}{1+\lambda_2}$. If we substitute the λ_2 formula obtained in Theorem 3, $\lambda_2 = \frac{\sigma^2}{\sigma_x^2}$, the actual shrinkage factor is given by the ratio $\frac{\sigma_x^2}{\sigma_x^2+\sigma^2}$, expressing the relation in energy between the clean signal and its noisy measurement.

When comparing the MAPS with the MAPC optimization tasks for a general dictionary, we observe that for low noise, $\sigma^2 \to 0$, the ℓ_2-term dominates the penalty function and the estimate is approximated mainly using this term, and the co-support Λ tends to be empty. On the other extreme, for high noise $\sigma^2 \to +\infty$, the determinant term dominates the penalty function and it is expected that $\dim\left(\mathbf{\Omega}_{\mathbf{\Lambda}}\right) \to d$, namely, $\hat{\mathbf{x}} \simeq \mathbf{0}$. Additionally, when $q \to 0$ the co-support probability term tends to $+\infty$ and the co-support Λ tends to be empty with an estimate $\hat{\mathbf{x}}$ given by the ℓ_2-term. In contrast, when $q \to 1$ the last term tends to $-\infty$ with a co-support Λ including all the rows in $\mathbf{\Omega}$. In such a case, the estimate is given by $\hat{\mathbf{x}} \simeq \mathbf{0}$. The deterministic problem with λ_2 and λ_0 given by Theorem 3 should have a similar behaviour. Thus, we expect to see a performance gap only for intermediate noise values.

We now run a numerical denoising experiment to show these differences. Our signals are small 2D images of size 4×3 samples, represented lexicographically as vectors in $\mathbb{R}^{12}$. Our analysis dictionary computes all the possible horizontal and vertical differences (without the borders), thus being of size $\mathbb{R}^{17\times 12}$. Using this dictionary, we draw 1000 signals as described in Section 3 with parameters $\sigma_x = 1$, $q = {}^{9}/_{17}$ and noise σ in the interval $[0.01, 1]$. Then, the various estimators are computed exhaustively iterating over each possible co-support. The performance is measured using the Relative-Mean-Squared-Error (RMSE) given by the formula $\frac{\|\mathbf{x}-\hat{\mathbf{x}}\|_2^2}{d\times\sigma^2}$, with a value lower than one referring to effective denoising. Then, we compare the RMSE of the estimators with that of the deterministic solution using two strategies for fixing λ_2, the first taken from Theorem 3, and

the second is an optimal value using an exhaustive search. Figure 1 shows the performance comparison for these methods. It can be seen that the deterministic approach (DET) using the Theorem 3 regularizer value performs similarly to the version with the optimized value. Moreover, while the MAPS and the DET approach have similar performances, the MAPC performance lags behind. As expected, the MMSE remains the best practical estimator (as it minimizes the MSE), while the Oracle estimator indeed seems to be a lower bound on the achievable performance.

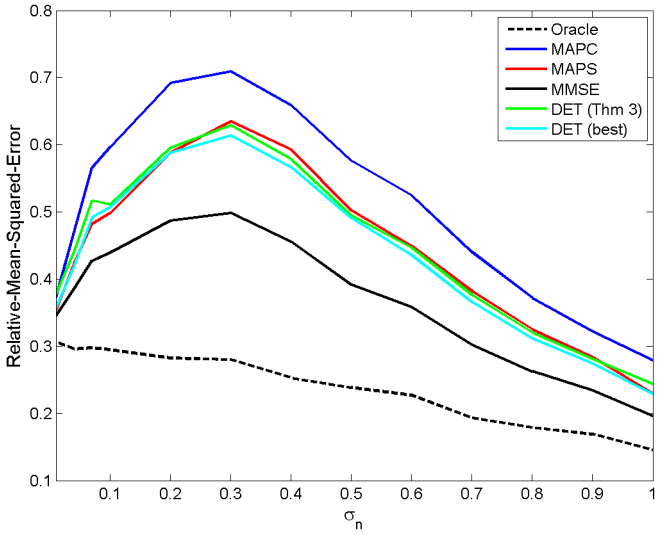


Figure 1: Comparison of denoising performance of the Bayesian estimators and the deterministic optimization task with the λ_0 closed formula from Theorem 3 and with the optimal λ_0 value ($\mathbf{\Omega}_{DIF} \in \mathbb{R}^{17\times 12}, q = 9/17, \sigma_x = 1, \sigma \in [0.01, 1]$).

6. Approximation Algorithms

For the MAP and MMSE estimators in Section 4, the optimization tasks involved are combinatorial in nature. All of them require a sweep over all possible 2^p co-supports to obtain the optimal solution. In order to overcome this complexity, we introduce approximation algorithms for these estimators. The MMSE estimate is approximated using a Gibbs sampler. An approximation to the MAPC and the MAPS estimators is computed using greedy pursuit algorithms, where for MAPS a special penalty approximation is required.

6.1. MMSE Approximation

The MMSE estimator in Section 4.2 requires the computation of an Oracle estimate for all possible co-supports Λ. This task is computationally prohibitive

so, instead, we sample some co-supports according to their probability, and compute their Oracle estimates. With this partial set of likely co-supports and their estimates, we approximate the MMSE estimate. This sampling scheme will use the core idea of the Gibbs sampler [23].

The sampling process begins with an initial co-support Λ^0, for instance an empty or a random co-support. Then, in every iteration the co-support Λ^i is obtained by updating one element (a row in $\mathbf{\Omega}$) from the previous co-support Λ^{i-1}. The updating of atom j is done by evaluating the probability of the atom to be part of the co-support based on the last sampled version. The probability of atom j to be active or inactive for the co-support Λ^i is given by

$$P\left(j|\Lambda_*^{i-1},\mathbf{y}\right)=\begin{cases}\frac{1}{t}\frac{1}{\sqrt{\det\left(\mathbf{C}_{\Lambda_*^{i-1}\cup\{j\}}\right)}}\exp\left\{-\frac{1}{2\sigma_x^2}\mathbf{y}^T\mathbf{C}^{-1}_{\Lambda_*^{i-1}\cup\{j\}}\mathbf{y}\right\}q & j\in\Lambda^i\\ \frac{1}{t}\frac{1}{\sqrt{\det\left(\mathbf{C}_{\Lambda_*^{i-1}}\right)}}\exp\left\{-\frac{1}{2\sigma_x^2}\mathbf{y}^T\mathbf{C}^{-1}_{\Lambda_*^{i-1}}\mathbf{y}\right\}(1-q) & j\notin\Lambda^i\end{cases}, \tag{31}$$

where Λ_*^{i-1} is the co-support Λ^{i-1} without element j, and t is a normalization factor. Then, we toss a coin based on the probabilities $P\left(j|\Lambda_*^{i-1},\mathbf{y}\right)$ to update the state for atom j in the co-support Λ^i. Together with the atom update, the estimate $\hat{\mathbf{x}}^i$ with the co-support Λ^i is computed. The process repeats by iterating through every atom in $\mathbf{\Omega}$ and continues cyclically after all the atoms are tested. After a certain number of iterations, the process is stopped and the MMSE is approximated as the average of the N estimates $\hat{\mathbf{x}}^i$ obtained:

$$\hat{\mathbf{x}}^{MMSE}=\frac{1}{N}\sum_{i=1}^{N}\hat{\mathbf{x}}^i.$$

The method is summarized in Algorithm 1.

6.2. MAPC Approximation

The optimization task in (16) for the MAPC estimator also suggests a sweep over all possible co-supports in order to find the optimal co-support that maximizes the penalty function. This penalty is given as

$$\frac{1}{2\sigma_x^2}\mathbf{y}^T\mathbf{C}_\Lambda^{-1}\mathbf{y}+\frac{1}{2}\ln\left[\det\left(\mathbf{C}_\Lambda\right)\right]-|\Lambda|\ln\left(\frac{q}{1-q}\right).$$

In this case, we propose a greedy pursuit algorithm to approximate a solution to task (16), following the idea in [17, 4].

The greedy pursuit algorithm for approximating MAPC starts from an empty co-support, i.e., $\Lambda^0=\emptyset$, and adds a new element (row) in each iteration of the algorithm. A new row is selected by iterating over every element j that is not in Λ^{i-1} and computing the value of the penalty function for a provisional co-support which involves the previous elements and j, i.e., $\Lambda_j=\Lambda^{i-1}\cup\{j\}$. Next, the row j_{min} that achieves the minimum penalty value is chosen to update the

Algorithm 1 MMSE Approximation

Input: Analysis dictionary $\mathbf{\Omega} \in \mathbb{R}^{p \times d}$, observation operator $\mathbf{M} \in \mathbb{R}^{k \times d}$, noisy observed signal $\mathbf{y} \in \mathbb{R}^d$, model parameters σ_x, σ, q, and N the number of iterations.

Output: Approximated signal $\hat{\mathbf{x}} \in \mathbb{R}^d$.

Initialization: Set $i = 0$, $\Lambda^0 := \emptyset$ (or any other initial co-support).

while $i < N$ **do**

  $i := i + 1$.

  Choose an Atom: $j \in \{l\}_{l=1}^{p}$

  Compute Provisional Co-support: $\Lambda_* := \Lambda^{i-1} - \{j\}$

  Compute Probabilities: Calculate probabilities $P(j|\Lambda_*, \mathbf{y})$ for atom j to be part of the co-support Λ^i (p_{in}) and to stay outside of it (p_{out}). Normalize the values such that $p_{in} + p_{out} = 1$.

  Toss a Coin: Draw a value p_{coin} in the range $[0 \dots 1]$.

  Update Co-support: for $p_{coin} < p_{in}$ update $\Lambda^i := \Lambda^{i-1} \cup \{\, j \,\}$, otherwise update $\Lambda^i := \Lambda^{i-1} - \{\, j \,\}$.

  Compute Current Estimate: $\hat{\mathbf{x}}^i := \mathbf{Q}_{\Lambda^i} \mathbf{M}^T \mathbf{C}_{\Lambda^i}^{-1} \mathbf{y}$.

end while

return $\hat{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \hat{\mathbf{x}}^i$.

Algorithm 2 MAPC and MAPS Approximations

Input: Analysis dictionary $\mathbf{\Omega} \in \mathbb{R}^{p\times d}$, sensing operator $\mathbf{M} \in \mathbb{R}^{k\times d}$, noisy signal $\mathbf{y} \in \mathbb{R}^k$, and parameters σ_x, σ and q.
Output: Signal $\hat{\mathbf{x}} \in \mathbb{R}^d$ with co-support $\hat{\Lambda}$.
Initialization: Set $i = 0$, $\Lambda^0 := \emptyset$, $F_1 = 1$ (for MAPS)
while Stopping criterion is not met **do**
  $i := i + 1$
  for $j \notin \Lambda^{i-1}$ **do**
    Update Provisional Co-support: $\hat{\Lambda}_j := \Lambda^{i-1} \cup \{j\}$
    Update Provisional Linear Dependencies: $\hat{\Lambda}_j := \hat{\Lambda}_j \cup \{k : \mathbf{Q}_{\hat{\Lambda}_j}\omega_k = \mathbf{0}\}$
    Provisional Error: With $\hat{\Lambda}_j$ as co-support, compute $Penalty(j)$ for MAPC using Equation (16), or for MAPS using Equation (20) with the last term approximated using Equation (32).
  end for
  Sweep: $j_{min} := \arg\min_{j\notin\Lambda^{i-1}} Penalty(j)$
  Update Co-support: $\Lambda^i := \Lambda^{i-1} \cup \{ j_{min} \}$
  Update Linear Dependencies: $\Lambda^i := \Lambda^i \cup \{k : \mathbf{Q}_{\Lambda^i}\omega_k = \mathbf{0}\}$
end while
return $\hat{\mathbf{x}} := \mathbf{Q}_{\hat{\Lambda}}\mathbf{M}^T\mathbf{C}_{\hat{\Lambda}}^{-1}\mathbf{y}$ and $\hat{\Lambda} = \Lambda^i$.

co-support, i.e., $\Lambda^i = \Lambda^{i-1} \cup \{j_{min}\}$. Additionally, rows that are linearly dependent on rows in the co-support are added to Λ^i. It is worth noting that these steps increment by one the dimension of the solution's subspace. The previous steps are repeated until a stopping criterion is met. Typical stopping criteria are given by the error term [12] or the co-sparsity of the co-support or its co-rank [17], or a local minimum in the penalty function for the updated co-support. After exiting the main loop, the approximated estimate $\hat{\mathbf{x}}$ is computed using the Oracle formula in (9) with the current co-support, see Algorithm 2.

It is worth noting that there is a similarity between the penalty function of the deterministic pursuit (1) (upto λ_2 and λ_0) and the penalty function of the MAPC estimator. This means that a similar approximation algorithm can be formulated for the deterministic pursuit. In fact, the pursuit algorithm that we propose is similar to the Backward Greedy (BG) and the Optimized Backward Greedy (OBG) algorithms proposed by Rubinstein et al.[11]. The main difference from these algorithms lies in the fact that the provisional error is computed using the penalty function and not only the ℓ_2-term like for BG or OBG. In many cases, this discrepancy produces different solutions. Additionally, while BG and OBG were proposed for the denoising task only, MAPC incorporates an observation matrix $\mathbf{M}$ for solving more general inverse problems.

6.3. MAPS Approximation

With a penalty function similar to that of MAPC, one may suggest a greedy pursuit approximation to Equation (20) for obtaining an approximation to the MAPS estimator. Unfortunately, the MAPS estimator requires knowledge of which co-supports span the same subspace. This extra piece of information is not easy to compute and it changes with the dependency relations between the rows of the dictionary $\mathbf{\Omega}$. Nevertheless, we present a greedy pursuit algorithm that approximates the MAPS estimator by approximating these dependency relations in an efficient manner.

Consider the summation in the last term of the MAPS optimization task in Equation (20) and assume that it has been fully computed for the previous iteration of a greedy pursuit algorithm with the subspace $\mathbf{\Psi}^{i-1}$, that is, assume that the following is known:

$$F_{i-1} = \sum_{\substack{\Phi \subseteq \Lambda_{\mathbf{\Psi}^{i-1}} \\ \text{span}\left(\mathbf{\Omega}_{\Phi}\right) = \mathbf{\Psi}^{i-1}}} \left(\frac{q}{1-q}\right)^{|\Phi|}.$$

How can this term be updated for $\mathbf{\Psi}^i$? When a new element is added to form the subspace $\mathbf{\Psi}^i$, all the subsets that span the same subspace $\mathbf{\Psi}^i$ should be taken into account. However, different linear dependencies may exist among the subsets of $\Lambda_{\mathbf{\Psi}^i}$, making this a non-trivial combinatorial problem. Therefore, we propose to approximate the original term by partial counting of these subsets. The idea is to easily count as many terms in the summation as we can, such that all these terms are correct (that is, they should be part of the true summation). Let us assume that a new row is added to the co-support, in the process of updating the subspace $\mathbf{\Psi}^i$ from $\mathbf{\Psi}^{i-1}$. Along with this newly added row, other rows may join as they are found to be linearly dependent on the resulting subspace - let us assume that the overall number of rows added is m. In order to obtain a co-support Φ that spans $\mathbf{\Psi}^i$, we may add any non-empty combination of these m elements to the previous co-support $\Lambda_{\mathbf{\Psi}^{i-1}}$[3]. If, for example, a specific subset of r such elements is added to the co-support, this should contribute $F_{i-1} \cdot \left(\frac{q}{1-q}\right)^r$ to the computation of F_i, as it corresponds to all the possible co-supports that were used in the summation of F_{i-1}, each with the extra r elements.

Thus, we obtain the approximation of the summation term by taking into account all the possible combinations with the m new elements by considering

[3]Recall that this is the maximal co-support that spans $\mathbf{\Psi}_{i-1}$.

all the possible choices of $1 \leq r \leq m$ out of m:

$$F_i = \sum_{\substack{\Phi \subseteq \Lambda_{\mathbf{\Psi}^i} \\ \text{span}\,(\mathbf{\Omega}_\Phi) = \mathbf{\Psi}^i}} \left(\frac{q}{1-q}\right)^{|\Phi|} \geq \left[\sum_{\substack{\Phi \subseteq \Lambda_{\mathbf{\Psi}^{i-1}} \\ \text{span}\,(\mathbf{\Omega}_\Phi) = \mathbf{\Psi}^{i-1}}} \left(\frac{q}{1-q}\right)^{|\Phi|}\right] \cdot \left[\sum_{r=1}^{m} \binom{m}{r} \left(\frac{q}{1-q}\right)^r\right]$$
$$= F_{i-1} \cdot \sum_{r=1}^{m} \binom{m}{r} \left(\frac{q}{1-q}\right)^r. \quad (32)$$

The reason for the inequality above is simple - there might be other co-supports that do not contain all $\Lambda_{\mathbf{\Psi}^{i-1}}$ and yet span $\mathbf{\Psi}_i$ and those evidently are not taken into account. When using only the equality part from the Inequality (32), one may suggest an approximate update formula between two iterations of a greedy algorithm for the summation term. Interestingly, the update formula depends only on the model parameter q and the number m of added rows. Instead of using the computation in Inequality (32), we can apply the Binomial theorem to obtain the following closed formula which is easier to compute:

$$\sum_{r=1}^{m} \binom{m}{r} \cdot \left(\frac{q}{1-q}\right)^r = \left(\frac{1}{1-q}\right)^m - 1.$$

The final greedy pursuit algorithm for MAPS approximation is presented in Algorithm 2.

6.4. Computational Complexity

We analyse the time complexity of the three approximation algorithms proposed in the previous sections as described in Appendix B. All the algorithms can be implemented using the Modified Gram Schmidt (MGS) algorithm and exploiting the matrix inversion lemma to reduce the complexity of inverting the matrix $\mathbf{C}_{\Lambda^i}$ and computing the provisional linear dependencies.

The MAPC and MAPS approximation algorithms behave similarly but with distinct penalty functions. More specifically, these functions differ only in their last term. Still the complexity in both cases is the same. The bottleneck in the time complexity of both algorithms is where the provisional linear dependencies are updated. The inner loop can be replaced by maintaining a p by p matrix $\mathbf{G^i}$, and thus reducing the complexity from $\mathcal{O}\left(p^2 d\right)$ - applying a matrix multiplication for each row - to $\mathcal{O}\left(p^2\right)$ to find the provisional linear dependencies. Since this matrix needs to be updated in each iteration, the main loop for both algorithms has $\mathcal{O}\left(p^2 r\right)$ time complexity; the outer loop gathering the co-support up to r dimensions (being r the co-rank, with $r < d$) adds the extra factor r. Additionally, during the initialization, the matrix $\mathbf{G^0}$ is computed with time

complexity $\mathcal{O}\left(p^2 d\right)$, as well as a Cholesky factorization of $\mathbf{C}_{\Lambda^0}$ that takes place with execution time proportional to $\mathcal{O}\left(k^3\right)$ to exploit the complexity improvements obtained by MGS. Under the assumption that $k \leq d$ (recall that k is the number of measurements in $\mathbf{y}$), the overall time complexity of both algorithms is $\mathcal{O}\left(p^2 d\right)$.

Comparing to the BG and OBG algorithms, the MAPC and MAPS approximations have the same complexity. The BG and OBG algorithms have time complexity $\mathcal{O}\left(p^2 d\right)$, also achievable exploiting the MGS method. The reason for the same complexity is that we manage to remove the inner loop in MAPC and MAPS by maintaining the extra matrix $\mathbf{G^i}$ and thus pairing the complexity to the BG, and OBG methods.

The time complexity of the approximation algorithm for MMSE estimation is bounded by the computation of the probabilities p_{in} and p_{out} of element j. When the co-support does not change between iterations, one of these probabilities remains the same and does not add to the complexity of the algorithm. The other probability depends of the current state in the co-support. If an element enters the co-support, then it is equivalent to computing a MGS step in $\mathcal{O}\left(pd\right)$. On the other hand, when an element leaves the co-support, all the other elements in the co-support should be updated to take into account the eliminated dimension. This update requires a restart of the MGS computation for all the elements in the co-support in $\mathcal{O}\left(pd^2\right)$ time. The main loop in this algorithm runs for N iterations, with an overall $\mathcal{O}\left(Npd^2\right)$ complexity in the worst case. Note that N should be at least p and typically $N \gg p$, because we must iterate at least one time per each element in the dictionary $\mathbf{\Omega}$. The initialization of the MMSE algorithm is the same as MAPC and MAPS because the same Cholesky factorization is computed in $\mathcal{O}\left(k^3\right)$ operations (with $k \leq d$). If the algorithm is initialized with a non-empty co-support then MGS should be applied for this co-support, adding $\mathcal{O}\left(p^2 d\right)$ to the complexity. All in all, the time complexity of the MMSE algorithm is bounded by $\mathcal{O}\left(Npd^2\right) \gg \mathcal{O}\left(p^2 d^2\right)$.

The solution $\hat{\mathbf{x}}$ of the MMSE estimate is not co-sparse, in contrast to the MAPC and MAPS solutions. Moreover, it is known that a Gibbs sampler may converge extremely slowly [50, 23] such that the number of iterations N is much bigger than p. We demonstrate this fact with the following denoising simulation. We select the dictionary $\mathbf{\Omega}_{DIF}$ for image patches of 8 by 8 pixels with $p = 112$ and $d = 64$. The MMSE approximation algorithm is tested with 1000 signals generated with the current model and with parameters $\sigma_x = 1, q = {}^{58}/_{112}$. The simulation is repeated for several noise values σ in the interval $[0.01, 1]$. The denoising performance is measured using RMSE. The graph in Figure 2 presents the average performance achieved after N iterations for several values of N. As expected, the performance improves with the number of iterations but the algorithm converges slowly. In this case, we note that the algorithm seems to converge when $N \approx dp$ for the current experiment. In general, the MMSE approximation has a greater time complexity of $\mathcal{O}\left(Nd\right)$ times the cost of the MAP-based approximations, making the later more affordable.

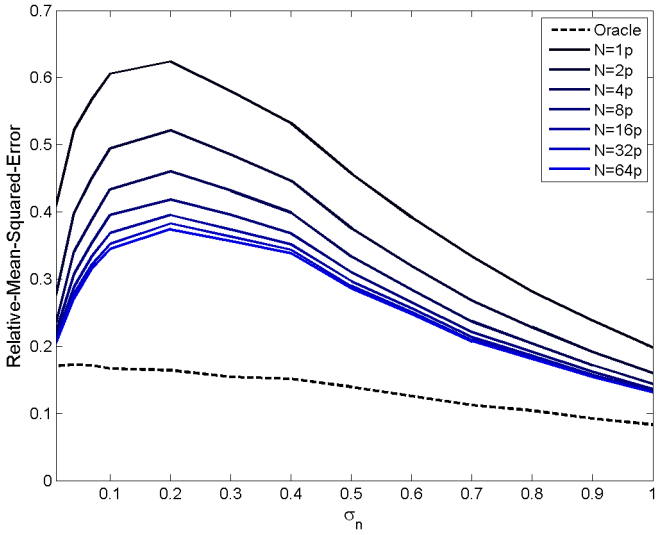


Figure 2: Denoising performance of the MMSE approximation with different values for the number of iterations N.

7. Simulation Results

In this section, we present a set of experimental results to evaluate the performance of the approximation algorithms. First, we present experiments on synthetic signals, demonstrating the ability of the approximation algorithms introduced in Section 6 to perform well compared to the optimal solutions. Second, we show image reconstruction results with the current algorithms applied to the denoising and inpainting problems. We present visual inpainting results for piecewise-constant and natural images. Note that in all these experiments we choose a specific analysis dictionary $\mathbf{\Omega}_{DIF}$ (horizontal and vertical derivatives) rather than optimizing it for best performance, as practiced in [17, 51, 52, 24]. Thus the shown results are not the best possible with the Analysis model, as our main goal here is to demonstrate the relevance of the Bayesian approach taken and the approximation methods proposed for practical inverse problems.

7.1. Synthetic Experiments

The first experiment shows the behavior of the approximation algorithms presented in Section 6 in comparison to their exhaustive counterparts, and this is done for the denoising task ($\mathbf{M} = \mathbf{I}$). In this experiment, the analysis dictionary $\mathbf{\Omega}_{DIF}$ contains horizontal and vertical finite differences for image patches of size 4 by 3 pixels ($p = 17, d = 12$). Recall that small dimensions are required in order to enable the computation of the exhaustive estimations. We generate 1000 signals (with $\sigma_x = 1, q = 9/17$) as described in Section 3 and apply the different methods for reconstruction. The values for the λ_0 and λ_2 regularizers

in the deterministic approach are given by the formulas in Theorem 3. The MAP-like and deterministic (DET) approximations stop when a local minimum is reached with respect to their penalty functions. The MMSE approximation runs for 400 (which is roughly 23p) iterations for each reconstructed signal. Figure 3 shows the average denoising performance for all these methods with the noise level σ varying in the range $[0.01, 1]$. The performance is measured using the Relative-Mean-Squared-Error (RMSE): $\frac{\|\mathbf{x}-\hat{\mathbf{x}}\|_2^2}{d\times\sigma^2}$. As can be seen, there is a good match between the approximate and exhaustive results.

The next synthetic experiment focuses on the inpainting problem – interpolating missing values in the signals. For this experiment, the analysis dictionary is $\mathbf{\Omega}_{DIF}$ as before. We generate 1000 signals as described in Section 3 (with $\sigma_x = 1, q = {}^{9}/_{17}, \sigma = 0.1$) and their respective observation matrices. An observation matrix $\mathbf{M}$ selects a randomly chosen subset of the samples from the signal while removing the rest. This matrix is built for each signal by randomly drawing rows from the identity matrix. The performance is measured using Root-Mean-Squared-Error (Root-MSE): $\sqrt{\frac{\|\mathbf{x}-\hat{\mathbf{x}}\|_2^2}{d}}$. In Figure 4, the various estimators and their approximations are shown as a function of the percentage of missing elements in the measured signal. Together with these methods, a trivial ℓ_2 solution[4] is included as a reference. The results show that the approximations are quite accurate, and the order between them is as expected.

The previous experiments require the computation of exact estimators by exhaustive sweeps through all possible co-supports. This is possible only for unrealistically low dimensional signals. Next, we run two more synthetic experiments, of denoising and inpainting, but this time using higher dimensional signals, and therefore we cannot compute these exact estimators. We select $\mathbf{\Omega}_{DIF}$ for image patches of size 8 by 8 pixels ($p = 112, d = 64$) as the analysis dictionary, as shown in Figure 7. Then, we generate 1000 signals as in the previous experiments with $\sigma_x = 1$, and $q = {}^{58}/_{112}$ as model parameters. The remaining parameters are the same as in the low dimensional experiments. The Gibbs sampler for MMSE estimation runs for $N = 32 \cdot p = 3584$ steps. Additionally, we included in the denoising plot the results for the known algorithms BG, OBG [17], and GAPn [12] to compare with the Bayesian estimators and the deterministic method. The error threshold used for these algorithms is set to $1.1\sqrt{d}\sigma$. We note that these algorithms do not assume a prior on $\mathbf{x}$ besides the co-sparsity property, thus the $\|\mathbf{x}\|_2^2$ term does not appear in their penalties. The results of these experiments are depicted in Figures 5 and 6, showing similar recovery behaviour as in the low-dimensional signals. The MAPS approximation gives better results than the deterministic approach in both experiments. Also, DET and MAPS achieved better performance than BG, OBG and GAPn. We recall that the difference between OBG and DET depends on how the provi-

[4]The trivial solution aims to solve the inverse ℓ_2 problem given by $\|\mathbf{y}-\mathbf{M}\mathbf{x}\|_2^2 + \lambda_2\|\mathbf{x}\|_2^2$. This solution is in fact the first step done by all the greedy algorithms presented in this work, when $\Lambda = \emptyset$.

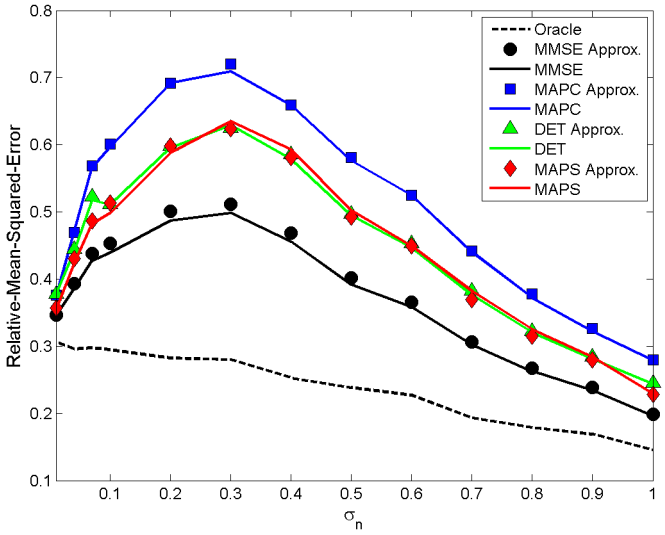


Figure 3: Denoising performance of the estimators and their respective approximation algorithms ($\mathbf{\Omega}_{DIF} \in \mathbb{R}^{17\times 12}, q = 9/17, \sigma_x = 1, \sigma \in [0.01, 1]$). The deterministic method uses λ_0 as given in Theorem 3.

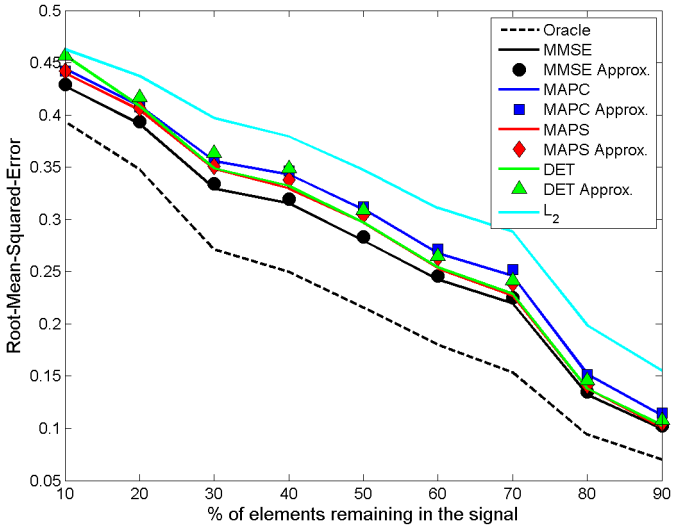


Figure 4: Inpainting results of the exhaustive Bayesian estimators (Oracle, MAPC, MAPS, and MMSE) and the deterministic pursuit (DET) compared to their approximated solutions as a function of the number of missing elements in the signal ($\mathbf{\Omega}_{DIF} \in \mathbb{R}^{17\times 12}, \sigma_x = 1, q = 9/17$).

sional error is computed. The OBG algorithm uses only the ℓ_2 term while DET also accounts for the ℓ_0 term for co-sparsity measure together with the ℓ_2 term.

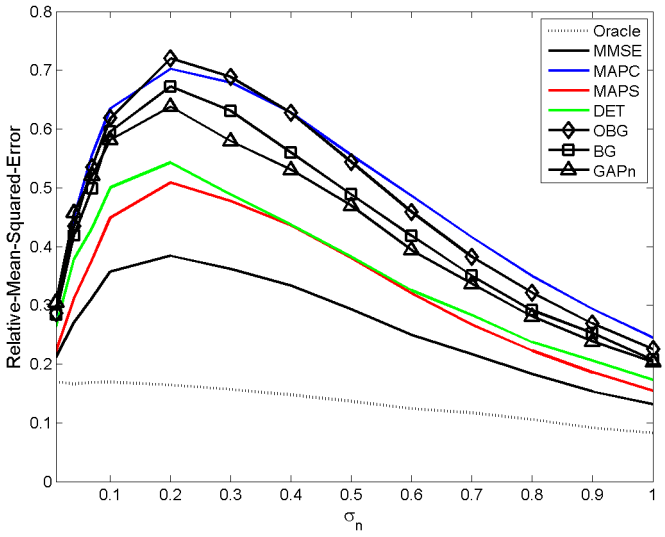


Figure 5: Comparison of the denoising performance for several approximation algorithms (Oracle, MAPC, MAPS, and MMSE), the deterministic approximation (DET), BG, OBG, and GAPn methods applied to high-dimensional signals (with $\mathbf{\Omega}_{DIF} \in \mathbb{R}^{112\times 64}, \sigma_x = 1, q = {}^{58}/_{112}$, $\sigma \in [0.01, 1]$).

7.2. Experiments with Images

We next present experimental results for denoising and inpainting of real images, in order to demonstrate the algorithms on true data. For the denoising task, we reconstruct a noisy version of Lena for various levels of white Gaussian noise with σ in the range $5-30$. In the denoising case, the observation operator $\mathbf{M}$ is the identity matrix $\mathbf{I}$. The approach we take is to reconstruct the image locally by extracting all possible 8×8 pixel patches with overlaps, denoising each such patch by imposing the Co-Sparse Analysis model and exploiting the above developed estimators, and then merging the results. Before processing each patch, we remove the mean value of the pixels in the patch in order to satisfy the zero mean assumption of the signal model, and it is then added back to the cleaned patches. Finally, the resulting patches are aggregated by simple averaging to reconstruct the full image.

The analysis dictionary $\mathbf{\Omega}_{DIF}$ used for this task is the same as in previous experiments (see Figure 7). The denoising performance is measured using RMSE, evaluated for the approximation algorithms for the Bayesian estimators and the deterministic pursuit approach (DET). The values for the model parameters for each estimator and regularizer parameters λ_0 and λ_2 for DET are exhaustively

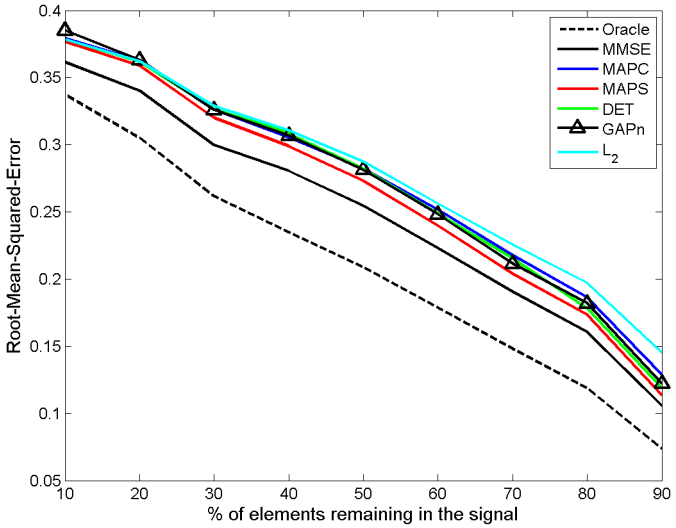


Figure 6: Comparison of inpainting results for several approximation algorithms (Oracle, MAPC, MAPS, and MMSE), the deterministic approximation (DET), and GAPn algorithm. The number of missing elements in the signal varies between 10% and 90%, the remaining parameters are $\mathbf{\Omega}_{DIF} \in \mathbb{R}^{112\times 64}, \sigma_x = 1, q = {}^{58}/_{112}$.

searched for best performance[5] and appear in Table 1. Additionally, the MMSE runs for $N = 16 \cdot p = 1792$ steps.

Figure 8 shows the performance of the various methods as a function of the noise level. We find that the Bayesian estimators and the Deterministic approach manage to reduce noise effectively. In particular, the MAP-based approaches and DET obtain similar results for all noise levels. Differences between

[5]One could consider finding these parameters via SURE, but this is beyond the scope of this work.

Noise σ	DET		MAPC		MAPS		MMSE	
	λ_2	λ_0	σ_x	q	σ_x	q	σ_x	q
5	0.010	7000	30	0.70	30	0.55	30	0.55
10	0.040	7000	50	0.75	40	0.60	40	0.60
15	0.046	15000	60	0.75	70	0.70	80	0.70
20	0.063	20000	80	0.75	80	0.70	80	0.70
25	0.037	60000	200	0.80	190	0.70	190	0.70
30	0.063	45000	200	0.80	200	0.70	190	0.70

Table 1: Algorithm parameters for denoising of Lena.

Noise σ	DET		MAPC		MAPS		MMSE	
	λ_2	λ_0	σ_x	q	σ_x	q	σ_x	q
5	0.0031	35000	200	0.65	200	0.55	200	0.55
10	0.0051	50000	200	0.60	200	0.55	200	0.55
15	0.0088	65000	200	0.65	200	0.55	200	0.55
20	0.0100	80000	200	0.60	200	0.50	200	0.60
25	0.0156	87500	200	0.60	200	0.50	200	0.60
30	0.0225	87500	200	0.65	200	0.50	200	0.65

Table 2: Algorithm parameters for inpainting of Lena with 75% missing pixels.

these Bayesian methods and the deterministic pursuit are seen in other general inverse problems when the observation matrix $\mathbf{M}$ is not the identity operator. On the other hand, the MMSE remains the best performer by far though its runtime complexity is much higher.

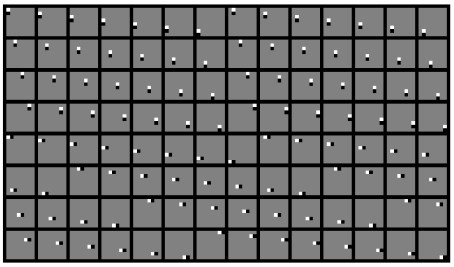


Figure 7: The atoms of the analysis dictionary $\mathbf{\Omega}_{DIF}$ of size 112 by 64. Each atom corresponds to a horizontal or vertical finite difference of a 8 by 8 pixels image patch.

The next experiment presents the performance for the inpainting task for various pursuit methods in a noisy environment. The setup for this problem is the same as in the previous denoising experiment, with an additional removal of 75% of the image pixels for the inpainting reconstruction. In this case, the observation matrix $\mathbf{M}$ varies for each patch depending on the missing pixels pattern. The inpainting performance is measured using Root-MSE. The experiment is run with the approximation algorithms for the Bayesian estimators and with the deterministic pursuit (DET) approach. Table 2 shows the model parameters for the estimators' approximations and the deterministic approach. These parameters are exhaustively searched for best performance. Similar as for denoising, the number of steps for MMSE is $N = 16 \cdot p = 1792$.

Figure 9 illustrates the Root-MSE results of the inpainting application as a function of the noise level. In contrast with the denoising results, we observe a performance gap between the deterministic pursuit (DET) and the Bayesian

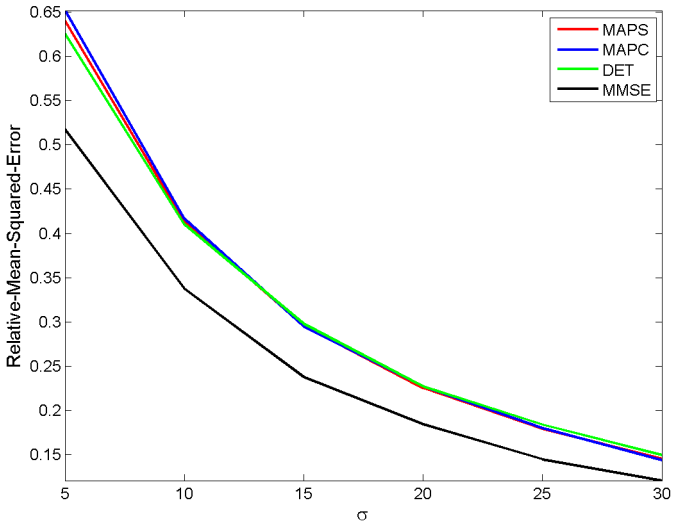


Figure 8: Comparison of denoising results with several algorithms for the Lena image as a function of the noise level.

estimators. This gap reveals the usefulness of the extra terms that are found in these estimators. The plot shows that the gap between MAPS and MAPC methods seen in the synthetic experiments has vanished in this experiment as well. Furthermore, MAPC achieves slightly better results for high noise levels. [6]

Next, we provide further the results for the inpainting application. Given an image, we remove 25% or 75% of the pixels and add white Gaussian noise with $\sigma = 20$ to the rest. We compare the inpainting performance of the approximation algorithms for the Bayesian estimators to the deterministic approach (λ_0 and λ_2 optimized for performance), and the GAPn method [12]. The analysis dictionary used for this task is $\mathbf{\Omega}_{DIF}$. We run the inpainting experiment for three images: Peppers, Lena, and a piecewise-constant (PWC) image. All the images have 256 by 256 pixels with a total of 62,001 patches. The parameters for the Bayesian estimators and the deterministic pursuit remains the same as above. The stopping threshold used in the GAPn is $1.1\sigma\sqrt{d}\cdot\sqrt{m}$, where m is the fraction of known elements in each patch.

The original and noisy images with 25% and 75% of the pixels missing are shown in Figure 10, the results for 25% missing pixels are presented in Figures

[6] These experiments include merging the patches after the reconstruction process, and thus there is an extra operator which does not belong to the model. We checked the average patch reconstruction performance of each algorithm before averaging and found similar results to those shown in Figure 9, and thus these graphs were not included.

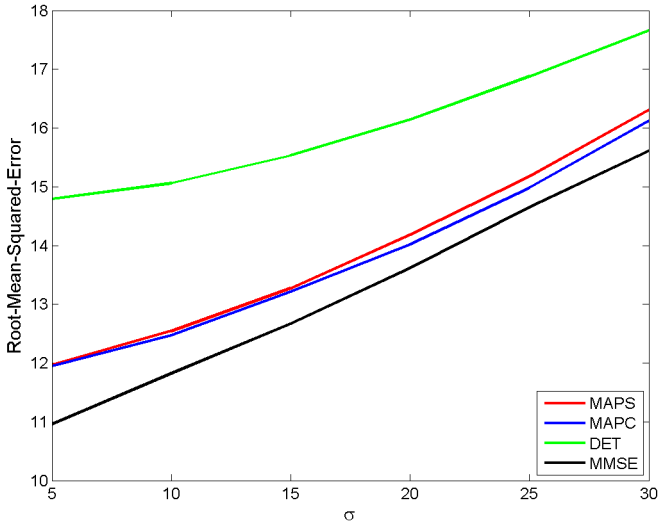


Figure 9: Comparison of inpainting results with several algorithms for the Lena image as a function of the noise level. The number of missing pixels in the image is 75%.

11, 12, and 13, while the results with 75% missing pixels in Figures 14, 15, and 16 for Peppers, Lena, and PWC images, respectively. The results shown include the Root-MSE performance for each method. The performances of the different approximations when 75% of the pixels are missing resemble those of the experiment above for all the images. On the other hand, MAPS obtains better results when 25% of the pixels are missing. Additionally, GAPn obtains results that are between the MAP-based estimators and the MMSE.

Visually, the algorithms succeeded in reconstructing the images, but with a few noticeable artifacts. A closer look reveals that all the analysis algorithms find it difficult to handle pixels with high added noise. The reason is that the removal of unusually high noise requires a higher cost in the ℓ_2-term than in the ℓ_0-term of the penalty functions. The number of these outlying pixels is relatively small, because the noise is normally distributed. Additionally, these methods yield images that tend to be sharp. This can be explained by the fact that in the Analysis model algorithms we used the dictionary $\mathbf{\Omega}_{DIF}$ that is well-suited for sharpening edges in images. We believe that all these results can be improved by selecting a more suitable dictionary or even learning it [17, 52, 24, 53].

8. Conclusions

In this work we have presented a generating model for co-sparse signals. We have derived Bayesian estimators based on this model: the Oracle estimator,

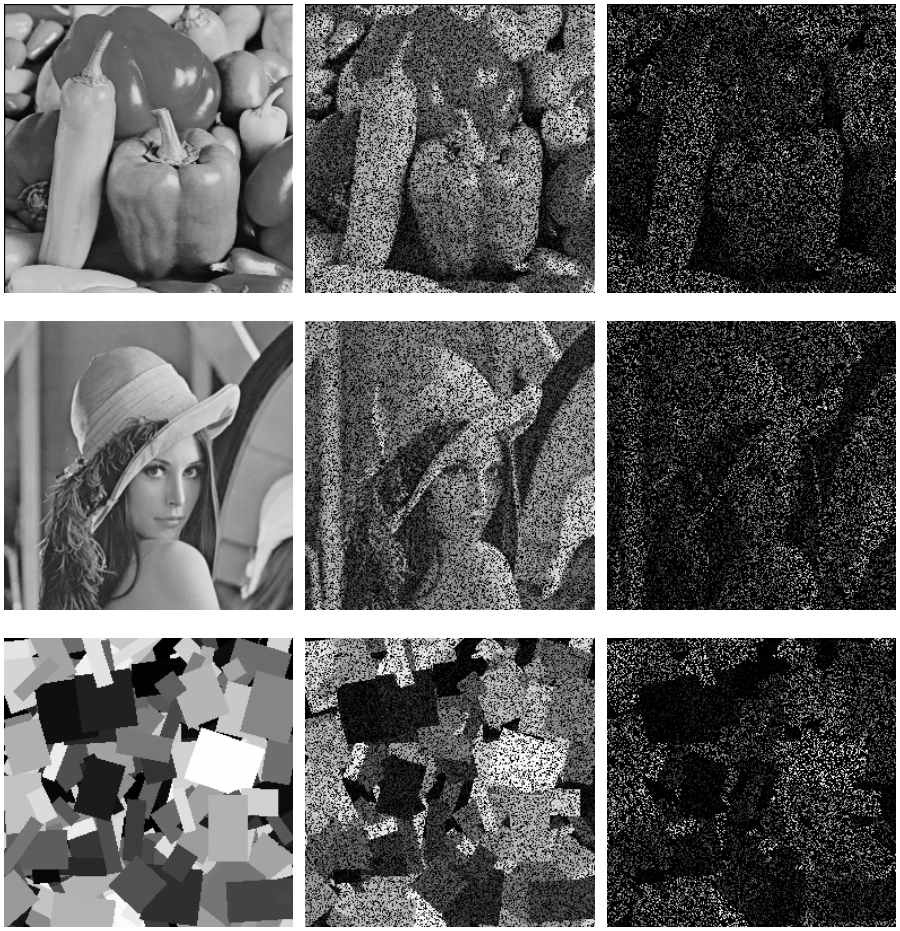
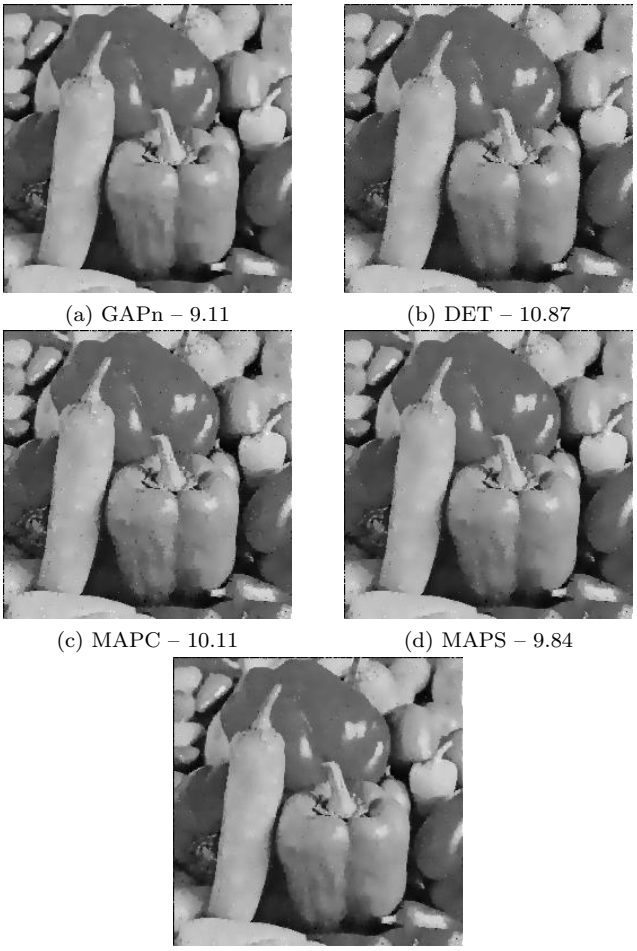


Figure 10: Original images (left), noisy images with 25% missing pixels and noise level $\sigma = 20$ (center), noisy images with 75% missing pixels and noise level $\sigma = 20$ (right) for inpainting of Peppers, Lena, and PWC images.



(a) GAPn – 9.11 (b) DET – 10.87

(c) MAPC – 10.11 (d) MAPS – 9.84

(e) MMSE – 9.00

Figure 11: Inpainting results of Peppers image with 25% missing pixels ($\sigma = 20$). The Root-MSE performance is shown next to each method.

(a) GAPn – 9.54

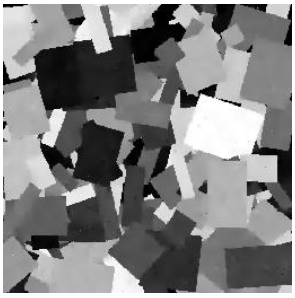
(b) DET – 11.16

(c) MAPC – 10.41

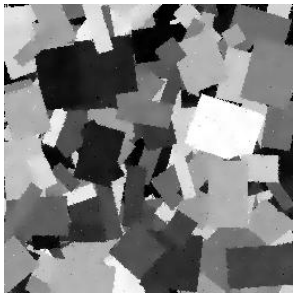
(d) MAPS – 10.25

(e) MMSE – 9.31

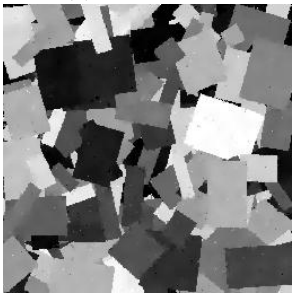
Figure 12: Inpainting results of Lena image with 25% missing pixels ($\sigma = 20$). The Root-MSE performance is shown next to each method.



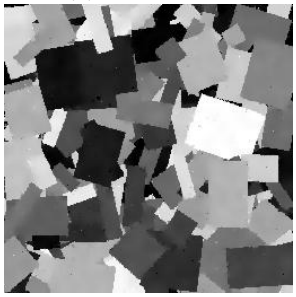
(a) GAPn – 9.73



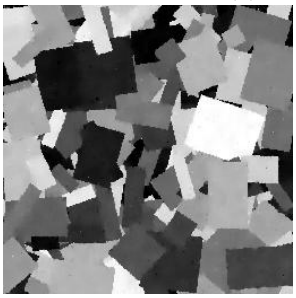
(b) DET – 12.65



(c) MAPC – 10.50

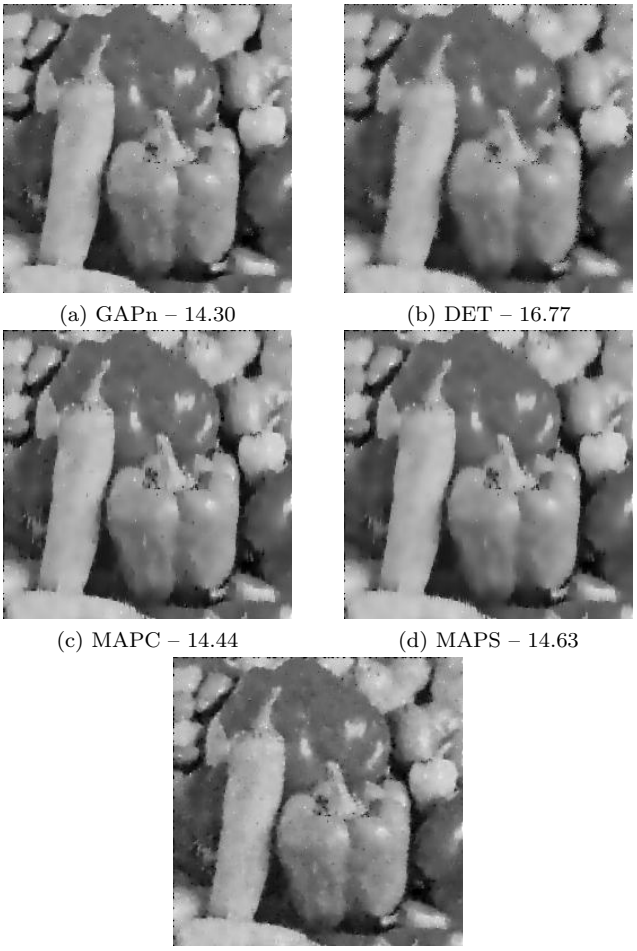


(d) MAPS – 10.26



(e) MMSE – 9.82

Figure 13: Inpainting results of PWC image with 25% missing pixels ($\sigma = 20$). The Root-MSE performance is shown next to each method.



(a) GAPn – 14.30 (b) DET – 16.77

(c) MAPC – 14.44 (d) MAPS – 14.63

(e) MMSE – 14.20

Figure 14: Inpainting results of Peppers image with 75% missing pixels ($\sigma = 20$). The Root-MSE performance is shown next to each method.

(a) GAPn – 14.39

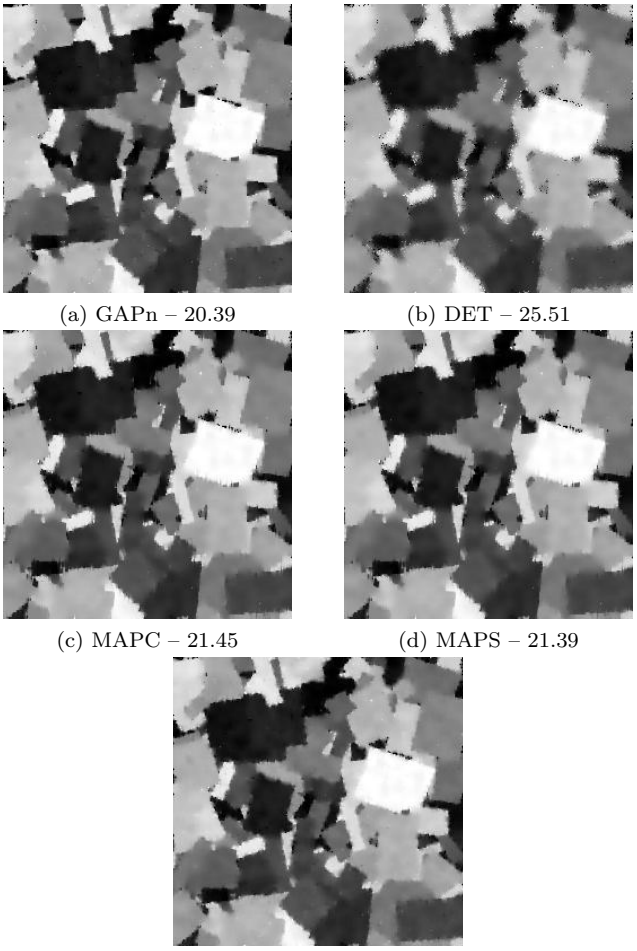
(b) DET – 16.14

(c) MAPC – 14.02

(d) MAPS – 14.19

(e) MMSE – 13.91

Figure 15: Inpainting results of Lena image with 75% missing pixels ($\sigma = 20$). The Root-MSE performance is shown next to each method.



(a) GAPn – 20.39 (b) DET – 25.51

(c) MAPC – 21.45 (d) MAPS – 21.39

(e) MMSE – 20.37

Figure 16: Inpainting results of PWC image with 75% missing pixels ($\sigma = 20$). The Root-MSE performance is shown next to each method.

the MMSE, and two versions of a MAP, the MAPC and the MAPS estimators. Furthermore, we have developed algorithms to approximate them and compared them to exact solutions. Additionally, we have shown the conditions for which the MAP-based estimators and the deterministic pursuit methods are fully-equivalent. Finally, we have demonstrated the capabilities of the methods in some basic experiments, showing effective recovery of synthetic and real image data.

We have shown that the deterministic pursuit and the MAP are different for the prior in Section 3 in cases where the assumptions in the theorem are not satisfied. However, additional exploration might lead to a prior where these problems are exactly the same for any inverse problem (with a general $\mathbf{M}$) and for any dictionary (with linear dependencies). Also, it would be interesting to extend the results obtained in this work with a theoretical study of the performance gaps that exist between the oracle and the estimators as well as the estimators and their approximated versions.

Appendix A. Proofs

Appendix A.1. Derivation of the Alternative Oracle Formula

We derive the equality $\mathbf{Q}_\Lambda \mathbf{M}^T \mathbf{C}_\Lambda^{-1} = \left(\frac{\sigma^2}{\sigma_x^2}\mathbf{I} + \mathbf{Q}_\Lambda \mathbf{M}^T \mathbf{M} \mathbf{Q}_\Lambda\right)^{-1} \mathbf{Q}_\Lambda \mathbf{M}^T$ used in the Oracle estimator in Equation (9). Recall that matrix $\mathbf{Q}_\Lambda = \mathbf{I} - \mathbf{U}_\Lambda \mathbf{U}_\Lambda^T$ is a projection matrix and it satisfies $\mathbf{Q}_\Lambda^2 = \mathbf{Q}_\Lambda$. Using this fact and applying the matrix inversion lemma to the matrix $\mathbf{C}_\Lambda = \frac{\sigma^2}{\sigma_x^2}\mathbf{I} + \mathbf{M}\mathbf{Q}_\Lambda \mathbf{M}^T$, we have

$$\begin{aligned}
\mathbf{Q}_\Lambda \mathbf{M}^T \mathbf{C}_\Lambda^{-1} &= \mathbf{Q}_\Lambda \mathbf{M}^T \left(\frac{\sigma^2}{\sigma_x^2}\mathbf{I} + \mathbf{M}\mathbf{Q}_\Lambda^2 \mathbf{M}^T\right)^{-1} \\
&= \mathbf{Q}_\Lambda \mathbf{M}^T \left[\frac{\sigma_x^2}{\sigma^2}\mathbf{I} - \frac{\sigma_x^4}{\sigma^4}\mathbf{M}\mathbf{Q}_\Lambda \left(\mathbf{I} + \frac{\sigma_x^2}{\sigma}\mathbf{Q}_\Lambda \mathbf{M}^T \mathbf{M} \mathbf{Q}_\Lambda\right)^{-1} \mathbf{Q}_\Lambda \mathbf{M}^T\right] \\
&= \frac{\sigma_x^2}{\sigma^2}\left[\mathbf{I} - \mathbf{Q}_\Lambda \mathbf{M}^T \mathbf{M} \mathbf{Q}_\Lambda \left(\frac{\sigma^2}{\sigma_x^2}\mathbf{I} + \mathbf{Q}_\Lambda \mathbf{M}^T \mathbf{M} \mathbf{Q}_\Lambda\right)^{-1}\right] \mathbf{Q}_\Lambda \mathbf{M}^T \\
&= \left(\frac{\sigma^2}{\sigma_x^2}\mathbf{I} + \mathbf{Q}_\Lambda \mathbf{M}^T \mathbf{M} \mathbf{Q}_\Lambda\right)^{-1} \mathbf{Q}_\Lambda \mathbf{M}^T.
\end{aligned}$$

Appendix A.2. Proof of the Determinant Identity

In Theorem 3, we use an identity of determinants to obtain the closed-form solution to the determinant of $\mathbf{C}_\Lambda$. Recall the identity in Equation (29),

$$\det\left(\mathbf{I}_{\mathbf{k}\times\mathbf{k}} + \frac{\sigma_{\mathbf{x}}^{\mathbf{2}}}{\sigma^{\mathbf{2}}}\mathbf{M}\mathbf{Q}_{\mathbf{\Lambda}}\mathbf{M}^{\mathbf{T}}\right) = \det\left(\mathbf{I}_{\mathbf{d}\times\mathbf{d}} + \frac{\sigma_{\mathbf{x}}^{\mathbf{2}}}{\sigma^{\mathbf{2}}}\mathbf{M}^{\mathbf{T}}\mathbf{M}\mathbf{Q}_{\mathbf{\Lambda}}\right). \tag{A.1}$$

Now we prove this identity for the general case. Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d \times k}$ be two matrices, then it follows that

$$\begin{aligned} \det\left(\mathbf{I_{k\times k}} + \mathbf{A^T B}\right) &= \det\left(\mathbf{I_{k\times k}} + \mathbf{B^T A}\right) = \\ \det\left(\mathbf{I_{d\times d}} + \mathbf{A B^T}\right) &= \det\left(\mathbf{I_{d\times d}} + \mathbf{B A^T}\right). \end{aligned} \tag{A.2}$$

The first equality is obtained from the determinant of the transpose of an invertible matrix $\mathbf{X}$ property, i.e., $\det(\mathbf{X}) = \det\left(\mathbf{X^T}\right)$. In order to prove the second equality, let us look at the following block matrix and decompose it into two possible pairs of triangular block matrices

$$\begin{aligned} \begin{pmatrix} \mathbf{I}_{d\times d} & -\mathbf{B} \\ \mathbf{A}^T & \mathbf{I}_{k\times k} \end{pmatrix} &= \begin{pmatrix} \mathbf{I}_{d\times d} & \mathbf{0} \\ \mathbf{A}^T & \mathbf{I}_{k\times k} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{d\times d} & -\mathbf{B} \\ \mathbf{0} & \mathbf{I}_{k\times k} + \mathbf{A}^T\mathbf{B} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{I}_{d\times d} & -\mathbf{B} \\ \mathbf{0} & \mathbf{I}_{k\times k} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{d\times d} + \mathbf{B}\mathbf{A}^T & \mathbf{0} \\ \mathbf{A}^T & \mathbf{I}_{k\times k} \end{pmatrix}. \end{aligned}$$

As these decompositions into block matrices are identical, then the determinants of the decompositions are the same. The determinant of a block triangular matrix is calculated as the product of the determinants of the diagonal blocks. Therefore, the determinant of the first matrix in both decompositions is 1 as $\det(\mathbf{I}) = 1$. In addition, we obtain

$$\begin{aligned} \det \begin{pmatrix} \mathbf{I}_{d\times d} & -\mathbf{B} \\ \mathbf{A}^T & \mathbf{I}_{k\times k} \end{pmatrix} &= \det\left(\mathbf{I_{k\times k}}\right) \det\left(\mathbf{I_{d\times d}} + \mathbf{A^T B}\right) \\ &= \det\left(\mathbf{I_{k\times k}} + \mathbf{B A^T}\right) \det\left(\mathbf{I_{d\times d}}\right), \end{aligned}$$

which shows the second equality. The last equality is obtained by using the determinant of a transpose matrix which completes the proof for Equation (A.2). The identity used to prove Theorem 3 is obtained by substituting $\mathbf{A}^T = \frac{\sigma_x^2}{\sigma^2}\mathbf{M}\mathbf{Q}_\Lambda$, and $\mathbf{B} = \mathbf{M}^T$ in Equation (A.2).

Appendix B. Optimizing the Approximation Algorithms

The algorithms presented in Section 6 require to compute the provisional linear dependencies, the inversion and determinant of matrix $\mathbf{C}_{\Lambda^i}$ in every iteration. Additionally, these algorithms update the co-support (and the solution) from the previous iteration. Therefore, the complexity of the provisional linear dependencies as well as the inversion and the determinant of $\mathbf{C}_{\Lambda^i}$ can be reduced using recursive formulas to update the results computed with the co-support Λ^{i-1} from the previous iteration.

Every time that an algorithm adds an element to the co-support, it may be linearly dependent on or independent of the other elements in the co-support. In the former case, the added element does not change the subspace of the solution as well as matrices $\mathbf{Q}_\Lambda$ and $\mathbf{C}_\Lambda$. In contrast, when the added element is linearly independent of the others, a new direction is added to the solution subspace and an update takes place in the projection matrix in Equation (3) and all its related

computations. Let Λ^{i-1} be the previous co-support and $\mathbf{u}_j$ be the new direction obtained from a Modified Gram-Schmidt (MGS) orthogonalization process from ω_j^T, the j^{th} row of the dictionary $\mathbf{\Omega}$. Then, the projection matrix in Equation (3) can be updated as follows

$$\mathbf{Q}_{\Lambda^i} = \mathbf{Q}_{\Lambda^{i-1}} - \mathbf{u}_j \mathbf{u}_j^T, \tag{B.1}$$

where the initial value is given by $\mathbf{Q}_{\Lambda^0} = \mathbf{I}$. With the recursive relation for matrix $\mathbf{Q}_{\Lambda^i}$, we obtain the following recursive formula for computing matrix $\mathbf{C}_{\Lambda^i}$ in Equation (7)

$$\mathbf{C}_{\Lambda^i} = \frac{\sigma^2}{\sigma_x^2}\mathbf{I} + \mathbf{M}\mathbf{Q}_{\Lambda^i}\mathbf{M}^T = \mathbf{C}_{\Lambda^{i-1}} - \mathbf{M}\mathbf{u}_j\mathbf{u}_j^T\mathbf{M}^T, \tag{B.2}$$

with an initial matrix $\mathbf{C}_{\Lambda^0} = \frac{\sigma^2}{\sigma_x^2}\mathbf{I} + \mathbf{M}\mathbf{M}^T$. We note that $\mathbf{M}\mathbf{Q}_\Lambda\mathbf{M}^T$ is a positive semidefinite matrix (a Gram matrix) and $\frac{\sigma^2}{\sigma_x^2}\mathbf{I}$ is a positive definite matrix, hence $\mathbf{C}_\Lambda$ is always invertible.

With a recursive formula for $\mathbf{C}_{\Lambda^i}$, we proceed with the determinant and matrix inversion formulas. We plug Equation (B.2) into the determinant of $\mathbf{C}_{\Lambda^i}$ and apply the matrix determinant lemma to obtain a recursive formula for computing the determinant,

$$\det\left(\mathbf{C}_{\mathbf{\Lambda^i}}\right) = \det\left(\mathbf{C}_{\mathbf{\Lambda^{i-1}}}\right) \cdot \left(1 - \mathbf{u}_j^T\mathbf{M}^T\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{u}_j\right). \tag{B.3}$$

The initial determinant value is obtained by computing the determinant of $\mathbf{C}_{\Lambda^0}$ defined above. In Equation (B.3), the previous determinant value and the inverted matrix $\mathbf{C}_{\Lambda^{i-1}}$ are needed. Nevertheless, we avoid to compute $\mathbf{C}_{\Lambda^{i-1}}^{-1}$ directly for every update of the co-support, as shown next.

Instead of computing the inverted matrix $\mathbf{C}_{\Lambda^i}^{-1}$ every iteration of the algorithms, we define a recursive formula to update every term that requires computing such an inversion. The recursive formulas for these terms are obtained by applying the matrix inversion lemma on $\mathbf{C}_{\Lambda^i}^{-1}$. We start with the ℓ_2-term in the penalty functions of the estimators,

$$\mathbf{y}^T\mathbf{C}_{\Lambda^i}^{-1}\mathbf{y} = \mathbf{y}^T\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{y} + \frac{\left(\mathbf{y}^T\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{u}_j\right)\left(\mathbf{u}_j^T\mathbf{M}^T\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{y}\right)}{1 - \mathbf{u}_j^T\mathbf{M}^T\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{u}_j}. \tag{B.4}$$

The initialization value given by $\mathbf{y}^T\mathbf{C}_{\Lambda^0}^{-1}\mathbf{y}$ still requires inverting the initial matrix $\mathbf{C}_{\Lambda^0}$. Note that the denominator in Equation (B.4) is exactly the same factor that appears in the determinant Equation (B.3). Next, we derive a recursive formula to compute the estimate $\hat{\mathbf{x}}^i$ as in Equation (9) with co-support Λ^i. For this purpose, we apply the matrix inversion lemma on $\mathbf{C}_{\Lambda^i}^{-1}$ and plug

the recursive formula for $\mathbf{Q}_{\Lambda^i}$ in Equation (B.1) into Equation (9) to obtain

$$\begin{aligned}\hat{\mathbf{x}}^i &= \mathbf{Q}_{\Lambda^i}\mathbf{M}^T\mathbf{C}_{\Lambda^i}^{-1}\mathbf{y} \\ &= \mathbf{Q}_{\Lambda^i}\mathbf{M}^T\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{y} + \mathbf{Q}_{\Lambda^i}\mathbf{M}^T\frac{\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{u}_j\mathbf{u}_j^T\mathbf{M}^T\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{y}}{1-\mathbf{u}_j^T\mathbf{M}^T\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{u}_j} \\ &= \hat{\mathbf{x}}^{i-1} - \mathbf{u}_j\mathbf{u}_j^T\mathbf{M}^T\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{y} + \mathbf{Q}_{\Lambda^i}\mathbf{M}^T\frac{\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{u}_j\mathbf{u}_j^T\mathbf{M}^T\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{y}}{1-\mathbf{u}_j^T\mathbf{M}^T\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{u}_j} \quad \text{(B.5)}\end{aligned}$$

where we need to update the matrix $\mathbf{Q}_{\Lambda^i}\mathbf{M}^T$ first, given the following formula $\mathbf{Q}_{\Lambda^i}\mathbf{M}^T = \mathbf{Q}_{\Lambda^{i-1}}\mathbf{M}^T - \mathbf{u}_j\mathbf{u}_j^T\mathbf{M}^T$. The initialization vector is computed by $\hat{\mathbf{x}}^0 = \mathbf{M}^T\mathbf{C}_{\Lambda^0}^{-1}\mathbf{y}$.

The recursive formulas (B.3), (B.4), and (B.5) suggest that $\mathbf{C}_{\Lambda^{i-1}}^{-1}$ should be computed and updated every iteration. However, the inverted matrix $\mathbf{C}_{\Lambda^{i-1}}^{-1}$ appears always as a matrix by vector multiplication, i.e., $\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{u}_j$. This is also the case for the term $\mathbf{M}\mathbf{u}_j$ in these formulas. Instead of computing these terms every iteration, we construct these terms for all the directions together and update them using recursive formulas. For this purpose, we compute a MGS step to obtain a new direction $\mathbf{u}_j$ for some iteration. This vector is computed by projecting the row ω_j^T of the dictionary $\mathbf{\Omega}$ into the subspace defined by the co-support Λ^{i-1}, and eventually normalizing the vector such that $\|\mathbf{u}_j\|_2 = 1$. As the MGS method is recursive in its nature, a copy $\mathbf{\Omega^i}$ with the orthogonalized versions of the dictionary $\mathbf{\Omega}$ rows is maintained. The matrix $\mathbf{\Omega^i}$ is updated every iteration applying a recursive MGS step: $\mathbf{\Omega^i}^T = \mathbf{\Omega^{i-1}}^T - \mathbf{u}_j\mathbf{u}_j^T\mathbf{\Omega^{i-1}}^T$. Similarly, $\mathbf{M}\mathbf{u}_j$ is obtained by maintaining the matrix $\mathbf{M}\mathbf{\Omega^i}^T$ and updating it using the same idea. Last, $\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{u}_j$ is obtained by building and updating the matrix $\mathbf{C}_{\Lambda^i}^{-1}\mathbf{M}\mathbf{\Omega^i}^T$ every iteration. This matrix requires a two step update process: (a) compute the term $\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{\Omega^i}^T$ for the updated matrix $\mathbf{\Omega^i}^T$ and (b) update the result using the formula $\mathbf{C}_{\Lambda^i}^{-1}\mathbf{M}\mathbf{\Omega^i}^T = \mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{\Omega^i}^T - \frac{\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{u}_j\mathbf{u}_j^T\mathbf{M}^T\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{\Omega^i}^T}{1-\mathbf{u}_j^T\mathbf{M}^T\mathbf{C}_{\Lambda^{i-1}}^{-1}\mathbf{M}\mathbf{u}_j}$ (obtained by using the matrix inversion lemma on $\mathbf{C}_{\Lambda^i}^{-1}$). We remark that these matrices don't have the normalizing factor involving the vector $\mathbf{u}_j$, such that $\|\mathbf{u}_j\|_2 = 1$. Therefore, when using columns from these matrices to compute the terms $\mathbf{C}_{\Lambda^i}^{-1}\mathbf{M}\mathbf{u}_j$ and $\mathbf{M}\mathbf{u}_j$, one should divide by the normalizing factor in the computations. Maintaining and updating these three matrices avoids inverting the matrix $\mathbf{C}_{\Lambda^i}$ every iteration, hence reducing the time complexity of the algorithms. The initialization values for these formulas still requires one matrix inversion, i.e., for $\mathbf{C}_{\Lambda^0}$, as well as computing its determinant. This is a k by k matrix that is computed using a Cholesky decomposition as it is a positive definite matrix. The Cholesky decomposition allows to compute the determinant straightforward as well as these matrices.

Additional to computing the inversion of matrix $\mathbf{C}_{\Lambda^i}$ and its determinant, the algorithms in Section 6 require to compute the provisional linear dependencies to decide which row should be added to the co-support Λ^i. As posed in

Algorithm 2, this can be achieved by iterating over each row j outside the co-support Λ^{i-1} and checking for linear dependencies of the provisional co-support $\hat{\Lambda}_j = \{j\} \cup \Lambda^{i-1}$. This set of provisional linear dependencies can be obtained by computing $\hat{\mathbf{\Omega}}_j^T = \mathbf{Q}_{\hat{\Lambda}_j} \mathbf{\Omega^T}$ and checking which columns are equal to zero. Note that this needs to be executed for each row not in the co-support Λ^{i-1} per every iteration of the outer loop iteration of the algorithm, becoming very expensive to compute. Multiplying the matrix $\mathbf{Q}_{\hat{\Lambda}_j}$ by $\mathbf{\Omega}^T$ is equivalent to orthogonalizing the rows in $\mathbf{\Omega}$. Using the recursive formula for $\mathbf{\Omega^i}^T$ described above, the same computation can be achieved only with matrix-by-vector multiplications by projecting the already orthogonalized elements in $\mathbf{\Omega^{i-1}}^T$ with a provisional direction $\mathbf{u}_j$: $\hat{\mathbf{\Omega}}_j^T = \mathbf{Q}_{\hat{\Lambda}_j} \mathbf{\Omega}^T = \mathbf{\Omega^{i-1}}^T - \mathbf{u}_j(\mathbf{u}_j^T \mathbf{\Omega^{i-1}}^T)$. Then the linear dependencies are the columns of $\hat{\mathbf{\Omega}}_j$ that become zero. In such a way, the cost is reduced by a factor of d for each row of the inner loop. Checking which columns of $\hat{\mathbf{\Omega}}_j^T$ are zero is equivalent to check which columns have zero norm. In particular, the ℓ_2-norm allows us to define a recursive formula for the norm of each column as follows:

$$\|\hat{\omega}_k\|_2^2 = \left\|\omega_k - \mathbf{u}_j \mathbf{u}_j^T \omega_k\right\|_2^2 = \|\omega_k\|_2^2 - (\mathbf{u}_j^T \omega_k)^2, \tag{B.6}$$

where $\hat{\omega}_k$ is the k^{th} column of $\hat{\mathbf{\Omega}}_j^T$, and ω_k is the k^{th} column of $\mathbf{\Omega^{i-1}}^T$. Recall that $\mathbf{u}_j$ is the column ω_j of $\mathbf{\Omega^{i-1}}^T$ after normalization to be a unit length vector. The inner loop can be eliminated computing Equation (B.6) simultaneously for all the directions $\mathbf{u}_j$ that are not in Λ^{i-1}. This is achieved by computing $\mathbf{G^{i-1}} = \mathbf{\Omega^{i-1}} \mathbf{\Omega^{i-1}}^T$ and dividing each resulting element by the respective normalizing factor. Even though we eliminate the inner loop in this way, computing this matrices multiplication does not reduce the complexity, yet. In order to reduce the computational cost, the following recursive formula is used:

$$\begin{aligned} \mathbf{G^i} &= \mathbf{\Omega^i} \mathbf{\Omega^i}^T \\ &= \mathbf{\Omega^{i-1}} \mathbf{\Omega^{i-1}}^T - (\mathbf{\Omega^{i-1}} \mathbf{u}_{j_{min}})(\mathbf{u}_{j_{min}}^T \mathbf{\Omega^{i-1}}^T) \\ &= \mathbf{G^{i-1}} - (\mathbf{\Omega^{i-1}} \mathbf{u}_{j_{min}})(\mathbf{u}_{j_{min}}^T \mathbf{\Omega^{i-1}}^T), \end{aligned}$$

where $\mathbf{G^0} = \mathbf{\Omega} \mathbf{\Omega}^T$, j_{min} is the element added to the co-support Λ^i in the iteration i, and $\mathbf{u}_{j_{min}}$ is the orthogonalized version of $\omega_{j_{min}}$. In this manner, all the inner products of the orthogonalized vectors are updated with every (outer) loop iteration. The norms of every $\hat{\omega}_k$ can be computed using matrix $\mathbf{G^i}$ and an the previous iteration norm value $\|\omega_k\|_2^2$.

The approximation algorithms presented in Section 6, can make extensive use of these recursive formulas. Every time that a new linearly independent element is added to the co-support, the previous formulas are useful. It is worth mentioning that the MMSE approximation algorithm may need to extract an element from the co-support for some specific cases. When this happens, all these matrices and terms must be changed to reflect this co-support modification. These changes require to compute the orthogonalization of the matrix

almost from scratch and may be quite expensive to compute, making the MMSE approximation time consuming.

References

[1] M. Elad, Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing, Springer, 2010.

[2] R. Baraniuk, V. Cevher, M. Wakin, Low-dimensional models for dimensionality reduction and signal recovery: A geometric perspective, Proceedings of the IEEE 98 (2010) 959–971.

[3] J.-L. Starck, F. Murtagh, J. M. Fadili., Sparse Image and Signal Processing, Cambridge University Press, 2010.

[4] S. Mallat, Z. Zhang, Matching pursuits with time-frequency dictionaries, IEEE Transactions on Signal Processing 41 (1993) 3397–3415.

[5] Y. Tsaig, D. L. Donoho, Compressed sensing, IEEE Transactions on Information Theory 52 (2006) 1289–1306.

[6] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, Image denoising by sparse 3-d transform-domain collaborative filtering, IEEE Transactions on Image Processing 16 (2007) 2080–2095.

[7] A. Buades, B. Coll, J. M. Morel, A non-local algorithm for image denoising, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, 2005, pp. 60–65.

[8] T. Peleg, Y. Eldar, M. Elad, Exploiting statistical dependencies in sparse representations for signal recovery, IEEE Transactions on Signal Processing 60 (2012) 2286–2303.

[9] S. Chen, D. Donoho, M. Saunders, Atomic decomposition by basis pursuit, SIAM Journal on Scientific Computing 20 (1998) 33–61.

[10] J. Tropp, Greed is good: algorithmic results for sparse approximation, IEEE Transactions on Information Theory 50 (2004) 2231–2242.

[11] M. Elad, P. Milanfar, R. Rubinstein, Analysis versus synthesis in signal priors, Inverse Problems 23 (2007) 947.

[12] S. Nam, M. Davies, M. Elad, R. Gribonval, Recovery of cosparse signals with greedy analysis pursuit in the presence of noise, in: 4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2011, pp. 361–364.

[13] S. Nam, M. Davies, M. Elad, R. Gribonval, The cosparse analysis model and algorithms, Applied and Computational Harmonic Analysis 34 (2013) 30–56.

[14] Y. Lu, M. Do, A theory for sampling signals from a union of subspaces, IEEE Transactions on Signal Processing 56 (2008) 2334–2345.

[15] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, Journal of the Royal Statistical Society: Series B 67 (2005) 301–320.

[16] S. Nam, M. Davies, M. Elad, R. Gribonval, Cosparse analysis modeling - uniqueness and algorithms, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011, pp. 5804–5807.

[17] R. Rubinstein, T. Peleg, M. Elad, Analysis K-SVD: A dictionary-learning algorithm for the analysis sparse model, IEEE Transactions on Signal Processing 61 (2013) 661–677.

[18] T. Peleg, M. Elad, Performance guarantees of the thresholding algorithm for the cosparse analysis model, IEEE Transactions on Information Theory 59 (2013) 1832–1845.

[19] R. Giryes, S. Nam, R. Gribonval, M. E. Davies, Iterative Cosparse Projection Algorithms for the Recovery of Cosparse Vectors, in: Proceedings of the 19th European Signal Processing Conference (EUSIPCO), Barcelona, Espagne, 2011, pp. 1460–1464.

[20] R. Giryes, M. Elad, Cosamp and SP for the cosparse analysis model, in: Proceedings of the 20th European Signal Processing Conference (EUSIPCO), 2012, pp. 964–968.

[21] E. J. Candès, Y. C. Eldar, D. Needell, P. Randall, Compressed sensing with coherent and redundant dictionaries, Applied and Computational Harmonic Analysis 31 (2011) 59–73.

[22] S. Vaiter, G. Peyré, C. Dossal, J. Fadili, Robust sparse analysis regularization, IEEE Transactions on Information Theory 59 (2013) 2001–2016.

[23] S. Geman, D. Geman, Stochastic relaxation, gibbs distributions, and the bayesian restoration of images, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6 (1984) 721–741.

[24] S. Hawe, M. Kleinsteuber, K. Diepold, Analysis operator learning and its application to image reconstruction, IEEE Transactions on Image Processing 22 (2013) 2138–2150.

[25] S. Hawe, M. Kleinsteuber, K. Diepold, Cartoon-like image reconstruction via constrained ℓ_p-minimization, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012, pp. 717–720. doi:10.1109/ICASSP.2012.6287984.

[26] M. Aharon, M. Elad, A. Bruckstein, K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation, IEEE Transactions on Signal Processing 54 (2006) 4311–4322.

[27] S. Roth, M. Black, Fields of experts, International Journal of Computer Vision 82 (2009) 205–229.

[28] J. Portilla, Image restoration through l0 analysis-based sparse optimization in tight frames, in: 16th IEEE International Conference Image Processing (ICIP), 2009, pp. 3909–3912.

[29] I. W. Selesnick, M. A. T. Figueiredo, Signal restoration with overcomplete wavelet transforms: comparison of analysis and synthesis priors, in: Proceedings of SPIE, volume 7446, 2009, pp. 74460D–74460D–15. doi:10.1117/12.826663.

[30] J. Wormann, S. Hawe, M. Kleinsteuber, Analysis based blind compressive sensing, IEEE Signal Processing Letters 20 (2013) 491–494.

[31] Q. Liu, S. Wang, L. Ying, X. Peng, Y. Zhu, D. Liang, Adaptive dictionary learning in sparse gradient domain for image recovery, IEEE Transactions on Image Processing 22 (2013) 4652–4663.

[32] Q. Ning, K. Chen, L. Yi, C. Fan, Y. Lu, J. Wen, Image super-resolution via analysis sparse prior, IEEE Signal Processing Letters 20 (2013) 399–402.

[33] E. Gil-Rodrigo, J. Portilla, D. Miraut, R. Suarez-Mesa, Efficient joint poisson-gauss restoration using multi-frame l2-relaxed-l0 analysis-based sparsity, in: 18th IEEE International Conference Image Processing (ICIP), 2011, pp. 1385–1388.

[34] A. Danielyan, V. Katkovnik, K. Egiazarian, Bm3d frames and variational image deblurring, IEEE Transactions on Image Processing 21 (2012) 1715–1728.

[35] I. Ram, I. Cohen, M. Elad, Patch-ordering-based wavelet frame and its use in inverse problems, Submitted to IEEE Transactions on Image Processing (2013).

[36] J.-F. Cai, S. Osher, Z. Shen, Split bregman methods and frame based image restoration, Multiscale Modeling & Simulation 8 (2010) 337–369.

[37] H. Ji, Y. Xu, Z. Shen, Wavelet based restoration of images with missing or damaged pixels, East Asian Journal on Applied Mathematics 1 (2011) 108–131.

[38] J.-F. Cai, H. Ji, Z. Shen, G.-B. Ye, Data-driven tight frame construction and image denoising, to appear in Applied and Computational Harmonic Analysis (2014).

[39] M. Ranzato, Y. Boureau, Y. LeCun, Sparse feature learning for deep belief networks, in: Advances in Neural Information Processing Systems 20 (NIPS), 2007, pp. 1185–1192.

[40] Y. Bengio, Learning deep architectures for AI, Foundations and Trends in Machine Learning 2 (2009) 1–127.

[41] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (2013) 1798–1828.

[42] H. Larochelle, Y. Bengio, J. Louradour, P. Lamblin, Exploring strategies for training deep neural networks, Journal of Machine Learning Research 10 (2009) 1–40.

[43] J. Turek, I. Yavneh, M. Elad, On MMSE and MAP denoising under sparse representation modeling over a unitary dictionary, IEEE Transactions on Signal Processing 59 (2011) 3526–3535.

[44] M. Protter, I. Yavneh, M. Elad, Closed-form mmse estimation for signal denoising under sparse representation modeling over a unitary dictionary, IEEE Transactions on Signal Processing 58 (2010) 3471–3484.

[45] M. Elad, I. Yavneh, A plurality of sparse representations is better than the sparsest one alone, IEEE Transactions on Information Theory 55 (2009) 4701–4714.

[46] P. Schniter, L. Potter, J. Ziniel, Fast bayesian matching pursuit, in: Information Theory and Applications Workshop, 2008, pp. 326–333.

[47] E. Larsson, Y. Selen, Linear regression with a sparse parameter vector, IEEE Transactions on Signal Processing 55 (2007) 451–460.

[48] S. Kay, Fundamentals of Statistical Signal Processing: Estimation Theory, volume I, Prentice Hall, 1993.

[49] G. Marsaglia, Conditional means and covariances of normal variables with singular covariance matrix, Journal of the American Statistical Association 59 (1964) 1203–1204.

[50] A. Raftery, S. Lewis, How many iterations in the Gibbs sampler, Bayesian statistics 4 (1992) 763–773.

[51] G. Peyré, J. Fadili, Learning analysis sparsity priors, in: 9th International Conference on Sampling, Theory and Applications (SAMPTA), 2011, p. 1.

[52] M. Yaghoobi, S. Nam, R. Gribonval, M. Davies, Noise aware rator learning for approximately cosparse signals, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012, pp. 5409–5412.

[53] M. Yaghoobi, S. Nam, R. Gribonval, M. Davies, Constrained overcomplete analysis operator learning for cosparse signal modelling, IEEE Transactions on Signal Processing 61 (2013) 2341–2355.

Vitae

Javier S. Turek received his B.Sc. and M.A. degrees from the Department of Computer Science, Technion, Israel, in 2007 and 2010, respectively. He received his M.A. degree in 2010 and is currently a Ph.D. Student in the Department of Computer Science at the Technion.

Since 2010, he has been a Ph.D. student with the Department of Computer Science, Technion, where he has studied sparse representations. His research interests include statistical signal and image processing, sparse representations, optimization and inverse problems.

Irad Yavneh received the Ph.D. degree in applied mathematics from the Weizmann Institute of Science, Rehovot, Israel, in 1991.

He is currently a Professor in the faculty of computer science at the TechnionIsrael Institute of Technology, Haifa, Israel. His research interests include multiscale computational techniques, scientific computing and computational physics, image processing and analysis, and geophysical fluid dynamics.

Michael Elad received his B.Sc. (1986), M.Sc. (1988) and D.Sc. (1997) from the department of Electrical engineering at the Technion, Israel.

Since 2003 Michael is a faculty member at the Computer Science department at the Technion, and since 2010 he holds a full-professorship position. Michael Elad works in the field of signal and image processing, specializing in particular on inverse problems, sparse representations and super-resolution. Michael received the Technions best lecturer award six times, he is the recipient of the 2007 Solomon Simon Mani award for excellence in teaching, the 2008 Henri Taub Prize for academic excellence, and the 2010 Hershel-Rich prize for innovation. Michael is an IEEE Fellow since 2012. He is serving as an associate editor for SIAM SIIMS, IEEE-TIT, and ACHA. Michael is also serving as a senior editor for IEEE SPL.