Overcomplete Dictionaries for
Sparse Representation of Signals

Michal Aharon
Overcomplete Dictionaries for Sparse Representation of Signals

Research Thesis Submitted in Partial Fulfillment of The Requirements for the Degree of Doctor of Philosophy.

Michal Aharon

Submitted to the Senate of the Technion - Israel Institute of Technology

KISLEV, 5767 HAIFA NOVEMBER 2006
I would like to thank my supervisor, Dr. Michael Elad, for his support and trust throughout the course of my PhD studies. His substantial and thorough approach, together with his genuine interest in the research subject, turned my research work into a great experience. A special thanks is also in order for Prof. Alfred Bruckstein, who accompanied me throughout my studies and was an active partner in this research.

The generous financial help of the Technion is gratefully acknowledged.

Finally, I would like to thank my husband, Hanoch, for being always beside me. This degree is dedicated to him with love.
vi
Contents

Chapter

1 Sparse Representation of Signals and Images 3
   1.1 Introduction and Symbols ............................. 3
   1.2 The Choice of the Dictionary .......................... 4
   1.3 The Sparseland Model for Signals ....................... 5
   1.4 The Contribution of this Work ......................... 6
   1.5 The Structure of this Report .......................... 9

2 Prior Art 11
   2.1 Sparse Coding ........................................ 11
      2.1.1 Orthogonal Matching Pursuit ....................... 12
      2.1.2 Basis Pursuit ..................................... 13
      2.1.3 FOCUSS ........................................... 14
   2.2 Theoretical Analysis of Pursuit Methods .............. 15
      2.2.1 Uniqueness of Sparse Representation .............. 16
      2.2.2 Evaluating the Pursuit Algorithm Results ........ 17
   2.3 Design of Dictionaries ................................. 19
      2.3.1 Generalizing the K-Means? ......................... 19
      2.3.2 Maximum Likelihood Methods ....................... 20
      2.3.3 The MOD Method .................................. 23
      2.3.4 Maximum A-posteriori Probability Approach ....... 24
      2.3.5 Unions of Orthonormal Bases ..................... 25
      2.3.6 Summary of the Prior Art ......................... 27

3 Uniqueness of the Dictionary 29
   3.1 Posing the Problem ................................... 29
   3.2 Statement of the Uniqueness Result ................. 29
   3.3 The Proof ............................................ 31
      3.3.1 Stage 1: Clustering The Signals .................. 32
      3.3.2 Stage 2: Detecting Pairs With Mutual Atom ...... 33
      3.3.3 Stage 3: Extracting the Mutual Atom ............ 34
3.3.4 Summarizing the Proof ........................................ 35
3.4 Summary .......................................................... 35

4 The K-SVD Algorithm ............................................... 37
  4.1 K-Means Algorithm for Vector Quantization ................. 37
  4.2 The K-SVD Algorithm ........................................... 39
    4.2.1 K-SVD — Generalizing the K-Means .................... 39
    4.2.2 K-SVD - Detailed Description ............................ 42
    4.2.3 From K-SVD Back to K-Means ............................. 48
    4.2.4 K-SVD - Implementation Details ....................... 49
    4.2.5 Synthetic Experiments ................................. 50
  4.3 Applications to Image Processing - Basic Results ........... 53
    4.3.1 Filling-In Missing Pixels ............................... 57
    4.3.2 Compression ............................................. 60

5 K-SVD-Based Image Denoising ...................................... 63
  5.1 Introduction .................................................... 63
  5.2 From Local to Global Bayesian Reconstruction .............. 67
    5.2.1 The Sparseland Model for Image Patches ............... 67
    5.2.2 From Local Analysis to a Global Prior ............... 68
    5.2.3 Numerical Solution ...................................... 70
  5.3 Example-Based Sparsity and Redundancy ....................... 71
    5.3.1 Training on the Corpus of Image Patches .............. 72
    5.3.2 Training on the Corrupted Image ...................... 73
  5.4 Results ....................................................... 75
  5.5 Summary ........................................................ 85

6 Constrained K-SVD .................................................. 89
  6.1 General Motivation ............................................ 89
  6.2 Non-Negative K-SVD (NN-K-SVD) .............................. 91
    6.2.1 General ................................................ 91
    6.2.2 Non-negative Decomposition ............................ 92
    6.2.3 Design of Non-Negative Dictionaries - Prior Art .... 93
    6.2.4 Non-Negative K-SVD - NN-K-SVD ...................... 94
    6.2.5 A Synthetic Experiment ................................. 96
  6.3 Shift-Invariant K-SVD (SI-K-SVD) ............................ 99
    6.3.1 General ................................................ 99
    6.3.2 Training Shift Invariant Dictionary - Prior Art .... 99
    6.3.3 K-SVD for Shift-Invariant Dictionaries (SI-K-SVD) .. 100
    6.3.4 SI-K-SVD - experiments ............................... 104
    6.3.5 Summary .............................................. 106
  6.4 Linear Constraints ............................................ 107
6.4.1 Linear Constraint K-SVD (LC-K-SVD) .................................. 108
6.4.2 Example - Multiscale Representation .................................. 109
6.5 Image Signature Dictionary .................................................. 119
6.5.1 Introduction ................................................................. 119
6.5.2 Training an Image Signature Dictionary ............................... 121
6.5.3 Experiments ................................................................. 124

7 Steps Towards a Geometrical Study of Sparseland .................... 129
7.1 Introduction and Motivation .................................................. 129
7.2 Representation Abilities of $L$-atoms ................................... 131
7.3 Representation Abilities of a Dictionary ................................. 137
  7.3.1 Representation by Non-Intersecting Sets of Atoms ............... 139
  7.3.2 Representation by Intersecting Sets of Atoms .................... 141
7.4 Summary ............................................................................. 146

8 Summary and Conclusions ...................................................... 149
8.1 General ............................................................................... 149
8.2 Future Work ....................................................................... 150

Bibliography ............................................................................ 153
### Tables

#### Table

5.1 Summary of the denoising PSNR results in [dB]. In each cell four denoising results are reported. Top left: results of Portilla et al. [87], Top right: overcomplete DCT, Bottom left: global trained dictionary, Bottom right: adaptive dictionary trained on noisy image. In each such set we highlighted the best result. All numbers are an average over 5 experiments. The last two columns present the average results over all images and their variance.

6.1 Average detection rates (in percentage) for the multiscale synthetic experiments. Notice the number of atoms are 9, 36 and 144 for the zero (coarsest), first and second level, respectively. Each experiment was performed 5 times.
## Figures

### Figure

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>The K-Means Algorithm</td>
<td>40</td>
</tr>
<tr>
<td>4.2</td>
<td>The K-SVD Algorithm</td>
<td>46</td>
</tr>
<tr>
<td>4.3</td>
<td>Synthetic results: for each of the tested algorithms and for each noise level, 50 trials were conducted, and their results sorted. The graph labels represent the mean number of detected atoms (out of 50) over the ordered tests in sets of 10 experiments.</td>
<td>52</td>
</tr>
<tr>
<td>4.4</td>
<td>A collection of 500 random blocks that were used for training, sorted by their variance.</td>
<td>54</td>
</tr>
<tr>
<td>4.5</td>
<td>The learned dictionary. The atoms are sorted in an ascending order of their variance, and stretched to maximal range for display purposes.</td>
<td>55</td>
</tr>
<tr>
<td>4.6</td>
<td>The overcomplete separable Haar dictionary.</td>
<td>55</td>
</tr>
<tr>
<td>4.7</td>
<td>The overcomplete DCT dictionary.</td>
<td>56</td>
</tr>
<tr>
<td>4.8</td>
<td>The RMSE for 594 new blocks with missing pixels using the learned dictionary, overcomplete Haar dictionary, and overcomplete DCT dictionary.</td>
<td>58</td>
</tr>
<tr>
<td>4.9</td>
<td>The corrupted image (left) with the missing pixels marked as points, and the reconstructed results by the learned dictionary, the overcomplete Haar dictionary, and the overcomplete DCT dictionary, respectively. The different rows are for 50% and 70% of missing pixels.</td>
<td>59</td>
</tr>
<tr>
<td>4.10</td>
<td>Compression results: Rate-Distortion graphs.</td>
<td>61</td>
</tr>
<tr>
<td>4.11</td>
<td>Sample compression results.</td>
<td>62</td>
</tr>
</tbody>
</table>

5.1 Denoising Procedure using a dictionary trained on patches from the corrupted image. For our experiments with noise level lower than $\sigma = 50$, we used the OMP pursuit method, and set $J = 10$, $\lambda = 30/\sigma$ and $C = 1.15$. | 76   |

5.2 The overcomplete DCT dictionary. | 79   |

5.3 The globally trained dictionary. | 80   |
5.4 Sample from the images used for training the global dictionary. 80
5.5 Comparison between the three presented methods (Overcomplete DCT, global trained dictionary and adaptive dictionary trained on patches from the noisy image) and the results achieved recently in [87] for three test images. 81
5.6 The improvement in the denoising results after each iteration of the K-SVD algorithm, executed on noisy patches of the image ‘Peppers’. 82
5.7 Example of the denoising results for the image ‘Barbara’ with σ = 20 – the original, the noisy, and two restoration results. 83
5.8 Example of the denoising results for the image ‘Barbara’ with σ = 20 – the adaptively trained dictionary. 84
5.9 The improvement (and later, deterioration) of the denoising results when increasing the value of λ in the averaging process in Equation (5.8). 86
5.10 The effect of changing the number of dictionary elements (K) on the final denoising results for the image ‘House’ and for σ = 15. 86

6.1 Finding a non-negative rank-1 approximation for a matrix \( A = dx^T \) 95
6.2 NN-K-SVD 97
6.3 On top, from left to right: True initial dictionary, K-SVD results after 60 iterations in the no-noise test, and after 100 iterations when the noise level was 15 SNR. On the bottom, from left to right: Hoyer’s algorithm results in the no-noise case after 600 and after 1500 iterations, and after 1500 iterations in the test with noise level of 15 SNR. 98
6.4 Illustration for approximation of a vector by a linear combination of 3 shifted versions of the atom \( \mathbf{d} \). 103
6.5 Each symbol represents the average detection rate of 5 tests. The tested methods were SI-K-SVD, ISL-DLA for shift invariant dictionaries, and the original K-SVD method (without shift invariance). 105
6.6 Detected atoms found by the SI-KSVD algorithm executed on 10 × 10 patches from natural images. The found atoms of size 4 × 4 are located in the center of each block, for visualization purposes. 106
6.7 MultiScale representation scheme. ↑ \( L_i \) or ↓ \( L_i \) represent enlarge or reduce size to level \( i \), \( A(\mathbf{D}^i) \) and \( S(\mathbf{D}^i) \) represent analysis and synthesis, respectively, using the \( i \)'th dictionary. 112
6.8 Typical execution of MS-KSVD (30dB). The graph presents the detection ratios of each level (the zero level is the coarsest one) for the first 6 iterations. 118
6.9 Samples from the 64×64 face images used for training the multiscale dictionaries. 119
6.10 Multiscale dictionaries extracted by the K-SVD algorithm, processed on real face images. The most left dictionary is the coarsest.

6.11 Illustration of several atoms from an ISD.

6.12 Image signature dictionary of size $100 \times 100$, trained on patches from real images.

6.13 Trained image signature dictionaries for several images. For each of these signatures we describe which image it is related to, the size of the signature, and the additive noise power that affected it.

6.14 Sample images we used for the various tests. From left to right: Barbara, House and Peppers.

6.15 Examining the shift-invariance property of the ISD.

7.1 Area calculation scheme. On the left - $n=2$ and $L=1$, on the middle - $n=3$ and $L=1$, and on the right - $n=3$, $L=2$.

7.2 Approximation of the function $\tan^{-1}$ using the identity function.

7.3 Calculation of the surface area of a dome.

7.4 Approximated and calculated ratio, with $n = 8$ and $L = 3$, and increasing error.

7.5 On the left, illustration of two overlapping strips. On the right, illustration of the bounding area of the intersection.

7.6 Illustration of the coverage of a Grassmanian dictionary of dimension 10 for several redundancy levels.
Abstract

In recent years there has been a growing interest in the study of sparse representation of signals. Using an overcomplete dictionary that contains prototype signal-atoms, signals are described as linear combinations of a few of these atoms. Applications that use sparse representation are many and include compression, regularization in inverse problems, feature extraction, and more.

Roughly speaking, the activity in this field concentrates around two main problems: (i) algorithms for performing sparse decomposition and their performance analysis, and (ii) dictionary composition methods. The ‘sparse decomposition’ activity deals with the pursuit problem - given a signal and an overcomplete dictionary, the goal is to find the smallest set of atoms from the dictionary to represent it. As this problem is known to be NP-hard, several approximation algorithms have been suggested, such as the matching pursuit, the basis pursuit, FOCUSS, and their variants. Recent work has been also directed at the study of these algorithms’ performance and prospects of success.

The construction of appropriate dictionaries that lead to the existence of sparse representations for a family of signals in mind is the second challenge addressed in recent years in this field. Among the various contributions on this problem, dictionary learning from examples is especially appealing as it can tune the found dictionary to specific and narrow families of signals.

In this work we concentrate on this dictionary learning process. Starting with a wide literature survey of pursuit techniques and existing dictionary learning
methods, this thesis turns to present a novel dictionary learning algorithm named the K-SVD, develops an analysis of its uniqueness properties, and explores ways to deploy the developed learning method to applications in image processing.

The problem setting for dictionary learning is the following - given a set of signals assumed to have a sparse representation over an unknown dictionary $D$, can we find $D$? A fundamental question that must be asked beforehand is whether $D$ is unique, or could there be several such dictionaries describing the same set of signals equally well. In our work we prove a uniqueness property, implying that in the quest for $D$ there is one ideal dictionary that we target.

At the heart of this work stands the K-SVD algorithm for learning dictionaries. We describe its development and analysis, and show some stylized applications to demonstrate its applicability and the advantages of trained dictionaries in general. Variations of the K-SVD algorithm for learning structural constrained dictionaries are also presented. Among those constraints are the non-negativity of the dictionary and shift invariance property. This work also includes a development of a state-of-the-art image denoising algorithm based on the K-SVD. This contribution is important as it strengthens the message that the general model of sparsity and redundancy, along with fitted dictionaries as considered here, could lead to the best known results in practical applications in image processing.

An underlying assumption throughout this thesis is that natural signals are represented well using a sparse linear composition of atoms from redundant dictionaries. We refer to this assumption as the "Sparseland" model, and provide an initial study of its geometrical behavior. More specifically, we provide bounds on the expected ratio of signals that can be represented by this dictionary with fixed sparsity constraints.
Chapter 1

Sparse Representation of Signals and Images

1.1 Introduction and Symbols

Recent years have witnessed a growing interest in the search for sparse representations of signals. Using an overcomplete dictionary matrix $D \in \mathcal{R}^{n \times K}$ that contains $K$ prototype signal-atoms for columns, $\{d_j\}_{j=1}^K$, a signal $y \in \mathcal{R}^n$ can be represented as a linear combination of these atoms. The representation of $y$ may either be exact $y = Dx$, or approximate, $y \approx Dx$, satisfying $\|y - Dx\|_p \leq \epsilon$. The vector $x \in \mathcal{R}^K$ contains the representation coefficients of the signal $y$. In approximation methods, typical norms used for measuring the deviation are the $\ell^p$-norms for $p = 1, 2$ and $\infty$. In this work we shall concentrate on the case of $p = 2$.

If $n < K$ and $D$ is a full-rank matrix, an infinite number of solutions are available for the representation problem, hence constraints on the solution must be set. The solution with the fewest number of nonzero coefficients is certainly an appealing representation. This sparsest representation is the solution of either

\[
(P_0) \quad \min_{x} \|x\|_0 \quad \text{subject to} \quad y = Dx,
\]
or

\[(P_{0,\varepsilon}) \min_x \|x\|_0 \text{ subject to } \|y - Dx\|_2 \leq \varepsilon,\]

where \(\|\cdot\|_0\) is the \(l^0\) norm, counting the nonzero entries of a vector.

Extraction of the sparsest representation is a NP-hard problem. Algorithms for finding approximating solutions have been extensively investigated and indeed, several effective decomposition algorithms are available. Thorough theoretical work studied the quality of these algorithms’ solution, in order to evaluate their similarity to the exact solutions. In all those methods, there is a preliminary assumption that the dictionary is known and fixed. In this research we address the issue of designing the proper dictionary, in order to better fit the sparsity model imposed.

1.2 The Choice of the Dictionary

An overcomplete dictionary \(D\) that leads to sparse representations can either be chosen as a pre-specified set of functions, or designed by adapting its content to fit a given set of signal examples.

Choosing a pre-specified transform matrix is appealing because it is simpler. Also, in many cases it leads to simple and fast algorithms for the evaluation of the sparse representation. This is indeed the case for overcomplete wavelets, curvelets, contourlets, steerable wavelet filters, short-time-Fourier transforms, and more. Preference is typically given to tight frames that can easily be pseudo-inverted. The success of such dictionaries in applications depends on how suitable they are to sparsely describe the signals in question. Multiscale analysis with oriented basis functions and a shift-invariant property are guidelines in such constructions.
In this research we consider a different route for designing dictionaries $D$ based on learning. Our goal is to find the dictionary $D$ that yields sparse representations for the training signals. We believe that such dictionaries have the potential to outperform commonly used pre-determined dictionaries. With ever-growing computational capabilities, computational cost may become secondary in importance to the improved performance achievable by methods which adapt dictionaries for special classes of signals.

### 1.3 The Sparseland Model for Signals

The concepts of sparsity and overcompleteness, together or separately, in representation of signals were proved to be highly effective. Applications that can benefit from them include compression, regularization in inverse problems, feature extraction, and more. Indeed, the success of the JPEG2000 coding standard can be attributed to the sparsity of the wavelet coefficients of natural images [76]. In denoising, wavelet methods and shift-invariant variations that exploit an overcomplete representation are among the most effective known algorithms for this task [30, 13, 96, 100]. Sparsity and overcompleteness have been successfully used for dynamic range compression in images [48], separation of texture and cartoon content in images [101, 102], inpainting [38], and more. Indeed, in this work we adopt the sparsity and redundancy concepts for achieving state-of-the-art denoising results.

For a signal family of interest (e.g. images), sparsity and redundancy as described above are imposed as a descriptive model. That is to say, we assume that each signal in this family has a (very) sparse representation over a specific and joint dictionary. We refer hereafter to this model as Sparseland. The Sparseland
model is defined by the following components:

- **Family of signals** - a set of signals with some mutual property. Different families may intersect or even include one another. For example, the set of facial-image-patches is a subset of real-image-patches. Clearly, the narrower the family of signals in mind, the better the Sparseland model handles its members.

- **Representation Error** - the maximal representation error we allow for each member in this family.

- **Sparsity constraint** - some constraint that forces sparsity of each coefficient vector. Such a constraint can be a distribution of the representations, or simply just the maximal number of non-zero entries allowed in each representation.

- **The Dictionary** - The representing dictionary $D$ that includes $K$ (often normalized) atoms.

The Sparseland model is the basis for all algorithms and applications described in this work, and it will accompany us throughout this report. We consider the quality of the results obtained in this work using this model as an empirical evidence that validates and strengthens the Sparseland model.

### 1.4 The Contribution of this Work

The field of sparse representations and the use of redundant dictionaries has been drawing a considerable attention lately. This interest emerges from both the practical impact of the wavelet theory on the signal/image processing community, and also the surprising recent theoretical results on the analysis of pursuit
methods. This emerging post-wavelet field, referred to herein as *Sparseland*, encompasses both theory and applications, in a way that appears to be gaining strength and interest.

In this thesis we concentrate on one of the most interesting and fundamental issues in this area - the design and use of data-driven dictionaries for sparse representations. Our main aim in this work is to demonstrate the effectiveness and potential of such tailored dictionaries, as also the simplicity of their design.

There are numerous questions one should consider, when approaching the problem of designing/training sparsifying dictionaries for signals. The most pressing of these problems is the issue of a feasible and simple algorithm for such a design. In our work we develop a novel such method, coined “the K-SVD”. This algorithm builds a dictionary that leads to sparse representations for the given set of training signals. The K-SVD is a straightforward generalization of the popular K-Means algorithm, used for Vector Quantization, and as such, it is easy to understand and to implement. Our work includes a comparison of this algorithm with competing methods, showing its efficiency and superiority. Using the K-SVD, the task that apparently seems the hardest—training the dictionary—becomes almost trivial and automatic.

If constraints are to be applied on the representing dictionary, variations of the K-SVD can be naturally and easily derived, due to the original simplicity of the algorithm’s setting. Indeed, in this work we show several such variations, among which are the non-negativity and shift invariance properties. The bottom line to this contribution is the fact that the K-SVD algorithm can be used with no “emotional involvement” in various applications.

Our first task, after putting forward the K-SVD algorithm, is an initial
demonstration of its potential to applications. This is demonstrated by applying it directly to several stylized applications in image processing and showing good results. The question remains, though, whether the Sparseland model and the K-SVD dictionary that serves it will be able to complete against the best available methods today in various applications. For this purpose we develop a full scheme for applying data-driven dictionaries for denoising of images. A simple local-based method is used in a way that creates a global prior for the whole image. This algorithm successfully competes and outperforms state-of-the-art image denoising methods, putting the K-SVD and the Sparseland model in the spotlight.

In a more theoretic discussion our work thoroughly dives into several interesting questions that concern sparse representation under data-driven dictionaries. Given a set of signals, in which each member was initially generated by sparse combinations over some unknown dictionary, we prove that this dictionary is necessarily unique, provided that enough signal examples are given. This proof provides a theoretic justification for our main concept - the quest for a sparse representing dictionary. In a different attempt to get a better understanding of the Sparseland model, this work also studies the geometrical structure dictated by sparse coding, concentrating on measuring the volume of signals that can be sparsely approximated by a given dictionary.

To summarize, in this thesis we try to demonstrate the simplicity and applicability of using data-driven dictionaries for sparse representation of signals, a promising area of research, which we believe has not yet reached the peak of its bloom.
1.5 The Structure of this Report

We start this report in Section 2 with reviewing the work that was done in the field of sparse representation and redundant dictionaries. First, several pursuit algorithms are described and a discussion concerning the evaluation of these algorithms’ performances is given. Then we describe the work that was done in the field of dictionary learning, and review the different approaches and algorithms.

The theoretic question concerning the uniqueness of the underlying dictionary is referred in Section 3. We prove that if a set of signals $Y$ is created by sparse combinations over some dictionary, under some mild conditions, this underlying dictionary is necessarily unique, and no other dictionary can represent this set of signals under the same sparsity constraints. This implies that in the quest for $D$ there is one ideal dictionary that we target.

In the center of this work stands a simple algorithm for dictionary design – the K-SVD. The K-SVD algorithm, generalizing the K-means algorithm for Vector Quantization, attempts to find the dictionary which can best represent all signals of interest sparsely. This algorithm is described and analyzed in detail in Section 4, along with several examples describing stylized applications in image processing.

The application of denoising images using learned dictionaries is revisited in Section 5, redeveloped and tuned in a way that leads to state-of-the-art denoising results. A limitation of the K-SVD is its ability to handle low-dimensional signals, which prevents its deployment to wide support images. The developed denoising algorithm described in this section uses the K-SVD locally on small image patches, and yet these local processes are merged into a global maximum-aposteriori prob-
ability estimation. Another intriguing feature of the proposed scheme is the fact that the dictionary is learned from the measured (noisy) image.

Several variations of the K-SVD for learning dictionaries with structural constraints are suggested and discussed in Section 6. The first constraint is non-negativity, applied on the dictionary entries, and possibly on the coefficients. K-SVD variation for shift invariant dictionaries is then presented. Here each dictionary element can serve as many different representing atoms, shifted variations of one another. The process of learning linear constrained dictionaries is presented afterwards, together with a suggestion for learning multiscale representation of images. Finally, a novel dictionary structure, referred to here as ‘Image Signature Dictionary’ (ISD) is suggested. The atoms of this dictionary are all available patches in some matrix, allowing cyclic shifts. The advantages of this scheme are discussed and an algorithm for training ISDs is presented.

In Section 7 we explore the sparseland model from a geometric point of view and confront the following problem. Given a dictionary $D \in \mathcal{R}^{n \times K}$, a sparsity constraint $L$ and a maximal allowed representation error $e$, a signal $y$ that can be represented is such for which there exists a subset of $L$ atoms from $D$ that approximate $y$ with an error less than $e$. The set of all represented signals is a subset of $\mathcal{R}^n$. The structure of this set is studied, with emphasis on its relative volume, in relation to the entire $\mathcal{R}^n$ space. We derive bounds on this size, and show that these bounds are strongly dependent on inner properties of $D$ such as the linear dependencies between its atoms.
Chapter 2

Prior Art

2.1 Sparse Coding

Sparse coding is the process of computing the representation coefficients, \( x \), based on the given signal \( y \) and the dictionary \( D \). This process, commonly referred to as “atom decomposition”, requires solving

\[
(P_0) \quad \min_x \|x\|_0 \quad \text{subject to} \quad y = Dx,
\]

or

\[
(P_{0,\epsilon}) \quad \min_x \|x\|_0 \quad \text{subject to} \quad \|y - Dx\|_2 \leq \epsilon,
\]

and this is typically done by a “pursuit algorithm” that finds an approximate solution, as exact determination of sparsest representations proves to be a NP-hard problem [15]. In this section we briefly discuss several such algorithms, and their prospects for success. Sparse coding is a necessary tool in designing dictionaries, hence it is important to have a good overview of methods for achieving it.

The simplest pursuit algorithms are the Matching Pursuit (MP) [75], Orthogonal Matching Pursuit (OMP) and Order Recursive Matching Pursuit (ORMP) [10, 16, 83, 104, 50]. These are greedy algorithms that select the dictionary atoms
sequentially. These methods are very simple, involving the computation of inner products between the signal and the dictionary atoms, and possibly deploying some least squares solvers or projections. Both (2.1) and (2.2) are easily addressed by changing the stopping rule of the algorithm.

A second well known pursuit approach is the Basis Pursuit (BP) [11]. It suggests a convexification of the problems posed in (2.1) and (2.2), by replacing the $\ell^0$-norm with an $\ell^1$-norm. The Focal Under-determined System Solver (FOCUSS) is very similar, using the $\ell^p$-norm with $p \leq 1$, as a replacement to the $\ell^0$-norm [52, 90, 89, 88]. Here, for $p < 1$ the similarity to the true sparsity measure is better, but the overall problem becomes non-convex, giving rise to local minima that may mislead in the search for solutions. Lagrange multipliers are used to convert the constraint into a penalty term, and an iterative method is derived based on the idea of iterated reweighed least-squares that handles the $\ell^p$-norm as an $\ell^2$ weighted norm.

### 2.1.1 Orthogonal Matching Pursuit

Orthogonal matching pursuit (OMP) is a greedy step-wise regression algorithm [75, 10, 16, 83, 104]. At each stage this method selects the dictionary element having the maximal projection onto the residual signal (with the assumption that the dictionary atoms have been normalized). After each selection, the representation coefficients w.r.t. the atoms chosen so far are found via least-squares. Formally, given a signal $\mathbf{y} \in \mathbb{R}^n$, and a dictionary $\mathbf{D}$ with $K$ $\ell^2$-normalized columns $\{\mathbf{d}_k\}_{k=1}^K$, we start by setting $\mathbf{r}_0 = \mathbf{y}$, $k = 1$, and perform the following steps:

1. Select the index of the next dictionary element $i_k = \text{argmax}_w |\langle \mathbf{r}_{k-1}, \mathbf{d}_w \rangle|$;
2. Update the current approximation $\mathbf{y}_k = \text{argmin}_{\mathbf{y}_k} \| \mathbf{y} - \mathbf{y}_k \|_2^2$, such that
\[ y_k \in \text{span} \{ \mathbf{d}_{i_1}, \mathbf{d}_{i_2}, \ldots, \mathbf{d}_{i_k} \}; \text{ and} \]

(3) Update the residual \( \mathbf{r}_k = \mathbf{y} - \mathbf{y}_k \).

The algorithm can be stopped after a predetermined number of steps, hence after having selected a fixed number of atoms. Alternatively, the stopping rule can be based on the norm of the residual, or on the maximal inner product computed in the next atom selection stage.

OMP is an appealing algorithm and very simple to implement. Unlike other methods, it can easily be programmed to supply a representation with an a priori fixed number of non-zero entries – a desired outcome in the training of dictionaries. There are several variants of the OMP that suggest (i) skipping the least-squares and using the inner product itself as a coefficient; (ii) applying least-squares per every candidate atom, rather than just using inner-products at the selection stage; (iii) projecting all non-selected atoms onto the space spanned by the selected atoms before each new atom selection; (iv) doing a faster and less precise search, where instead of searching for the maximal inner product, a nearly maximal one is selected, thereby speeding up the search; and there are more.

2.1.2 Basis Pursuit

The Basis Pursuit (BP) algorithm [11] proposes the replacement of the \( \ell^0 \)-norm in (2.1) and (2.2) with an \( \ell^1 \)-norm. Hence solutions of

\[
(P_1) \; \min_{\mathbf{x}} \| \mathbf{x} \|_1 \; \text{subject to} \; \mathbf{y} = \mathbf{Dx},
\]

in the exact representation case, and

\[
(P_{1,\epsilon}) \; \min_{\mathbf{x}} \| \mathbf{x} \|_1 \; \text{subject to} \; \| \mathbf{y} - \mathbf{Dx} \| \leq \epsilon,
\]
in the approximate one, lead to the BP representations. Solution of (2.3) amounts to linear programming and efficient solvers for such problems exist. The approximate form (2.4) leads to a quadratic programming structure, and again, there exist efficient solvers for such problems. Recent work on iterated shrinkage algorithms provide highly efficient and tailored methods for minimizing (2.4) in a setting which resembles the OMP (see [35, 37, 44, 45, 14]).

### 2.1.3 FOCUSS

FOCUSS stands for FOCal Underdetermined System Solver [52, 90, 89]. This is an approximating algorithm for finding the solution of either (2.1) or (2.2), by replacing the $\ell^0$-norm with an $\ell^p$ one for $p \leq 1$.

For the exact case problem, $(P_0)$, this method requires solving

$$(P_p) \quad \min_{\mathbf{x}} \|\mathbf{x}\|_p^p \text{ subject to } \mathbf{y} = \mathbf{Dx}. \tag{2.5}$$

The use of a Lagrange multiplier vector $\lambda \in \mathbb{R}^n$ here yields the Lagrangian function

$$\mathcal{L}(\mathbf{x}, \lambda) = \|\mathbf{x}\|_p^p + \lambda^T (\mathbf{y} - \mathbf{Dx}). \tag{2.6}$$

Hence necessary conditions for a pair $\mathbf{x}, \lambda$ to be a solution of (2.5) are

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) = p\Pi(\mathbf{x})\mathbf{x} - \mathbf{D}^T \lambda = 0 \quad \text{and} \quad \nabla_{\lambda} \mathcal{L}(\mathbf{x}, \lambda) = \mathbf{Dx} - \mathbf{y} = 0, \tag{2.7}$$

where we have defined $\Pi(\mathbf{x})$ to be a diagonal matrix with $|x_i|^{p-2}$ as its $(i, i)^{th}$ entry. The split of the $\ell^p$-norm derivative into a linear term multiplied by a weight matrix is the core of the FOCUSS method, and this follows the well-known idea of iterated reweighed least-squares [64]. Several simple steps of algebra lead to the solution

$$\mathbf{x} = \Pi(\mathbf{x})^{-1} \mathbf{D}^T (\mathbf{D}\Pi(\mathbf{x})^{-1}\mathbf{D}^T)^{-1} \mathbf{y}. \tag{2.8}$$
While it is impossible to get a closed form solution for $x$ from the above result, an iterative replacement procedure can be proposed, where the right hand side is computed based on the currently known $x_{k-1}$, and this leads to the updating process,

$$x_k = \Pi(x_{k-1})^{-1}D^T(D\Pi(x_{k-1})^{-1}D^T)^{-1}y.$$  

A regularization can, and should be introduced [88] to avoid near-zero entries in the weight matrix $\Pi(x)$.

For the treatment of $(P_0,\epsilon)$ via the $(P_p,\epsilon)$ parallel expressions can be derived quite similarly, although in this case the determination of the Lagrange multiplier is more difficult, and must be searched within the algorithm [88].

Both the BP and the FOCUSS can be motivated based on Maximum A-posteriori Probability (MAP) estimation, and indeed several works used this reasoning directly [71, 82, 81, 72]. The MAP can be used to estimate the coefficients as random variables by maximizing the posterior $P(x|y,D) \propto P(y|D,x)P(x)$. The prior distribution on the coefficient vector $x$ is assumed to be a super-Gaussian iid distribution that favors sparsity. For the Laplace distribution this approach is equivalent to BP [71].

### 2.2 Theoretical Analysis of Pursuit Methods

Extensive study in recent years of the pursuit algorithms has established that if the sought solution, $x$, is sparse enough, these techniques recover it well in the exact case [29, 36, 26, 54, 47, 104]. Further work considered the approximated versions and has shown stability in recovery of $x$ [28, 105]. The recent front of activity revisits those questions within a probabilistic setting, obtaining more realistic assessments on pursuit algorithms performance and success [24, 25, 6].
The properties of the dictionary $D$ set the limits that may be assumed on sparsity. In this section we briefly mention the main results of this study.

## 2.2.1 Uniqueness of Sparse Representation

For the analysis of the uniqueness of a sparse representation, as well as for the study of pursuit algorithms’ performance, we need to define two measures of quality for the dictionary $D$, the *Mutual Coherence* and the *Spark*. The *Mutual Coherence* of a dictionary $D$, denoted by $\mu(D)$, is defined as the maximal absolute scalar product between two different normalized atoms of $D$,

$$\mu(D) = \max_{i \neq j} |d_i^T d_j|.$$ (2.10)

The mutual coherence of a dictionary measures the similarity between the dictionary’s atoms. For an orthogonal matrix $D$, $\mu(D) = 0$. For an overcomplete matrix ($K > n$) we necessarily have $\mu(D) > 0$. As we shall see next, there is an interest in dictionaries with $\mu(D)$ as small as possible for sparse representation purposes. If $\mu(D) = 1$, it implies the existence of two parallel atoms, and this causes ambiguity in the construction of sparse atom compositions. In [103] it was shown that for a full rank dictionary of size $n \times K$

$$\mu \geq \sqrt{\frac{K - n}{n(K - 1)}},$$ (2.11)

and equality is obtained for a family of dictionaries called Grassmannian frames. For $K \gg n$ the mutual coherence we can expect to have is thus of the order of $1/\sqrt{n}$.

The *Spark* of a dictionary $D$ is the smallest number of columns that form a linearly dependent set [26]. Despite the similar definition, note that spark is markedly different from the matrix rank, being the greatest number of linearly
independent columns. A trivial relation between the spark $\sigma\{D\}$ and the mutual coherence $\mu\{D\}$ is [26]

$$\sigma\{D\} \geq 1 + \frac{1}{\mu\{D\}}.$$  \hspace{1cm} (2.12)

Referring to the problem posed in (2.1), we first quote a result that poses a condition on its solution $x$ such that it guarantees uniqueness:

**Theorem 1:** (see [26]): A linear representation over $m$ atoms (i.e., $\|x\|_0 = m$) is unique if

$$m < \frac{\sigma\{D\}}{2}. \hspace{1cm} (2.13)$$

The work in [28] generalizes this theorem for the solution of (2.2). It shows that, while exact uniqueness cannot be guaranteed, an approximate one that allows a bounded deviation can be claimed.

### 2.2.2 Evaluating the Pursuit Algorithm Results

The uniqueness theorem mentioned above implies that a representation with less than $\sigma\{D\}/2$ non zero elements is the sparsest available one, and therefore, if a pursuit algorithm retrieves such a solution it is necessarily the one desired. However, in which cases may we expect the pursuit algorithms to retrieve this exact solution, and when can we guarantee their success? Those kind of questions, concerning the connection between the pursuit algorithm’s results and the true solutions to (2.1) or (2.2) has been studied extensively in recent years. The following is a central result on the expected behavior of the MP and BP methods for sparse enough solutions:

**Theorem 2:** (see [26, 54, 104, 47]): If the sought solution, $x$, for the problem
(2.1), satisfies
\[ \|x\|_0 < \frac{1}{2} \left(1 + \frac{1}{\mu(D)}\right) \] (2.14)
then both BP and MP will recover it exactly.

This result suggests that for representations with less than \( O(\sqrt{n}) \) non-zeros, pursuit methods can succeed. Results of similar nature and strength were developed for structured dictionaries, constructed as a concatenation of several unitary matrices \([36, 54, 27]\). Such a dictionary structure is restrictive, but decomposition under this kind of dictionaries can be done efficiently using the Block Coordinate Relaxation (BCR) method \([93]\), being a fast variant of the BP.

Recent work by Gribonval and Nielsen analyzed the \((P_p)\) problem and showed its equivalence to \((P_0)\), under conditions similar in flavor to the sparsity conditions mentioned above \([53]\). Hence, the FOCUSS approach too enjoys the support of some theoretical justification, like the two discussed beforehand. However, the analysis says nothing about local minima traps and prospects in hitting those in the FOCUSS-algorithm.

Other work considered the approximated version (2.2) and showed stability in recovery of \(x\), meaning that pursuit algorithms lead to a solution in the proximity of the true optimal one. This research concerns the connection between the \((P_{0,\epsilon})\) and \((P_{1,\epsilon})\), and is reported in \([28, 105, 27]\).

All the above results consider the worst case scenario, and the bounds derived are therefore too pessimistic. The pursuit algorithms are known (empirically) to succeed in recovering sparse representations even when the number of non zero elements is substantially beyond those bounds. Indeed, the recent front of theoretical activity in this field revisits the above questions from a probabilistic point of view, obtaining more realistic assessments on pursuit algorithms perfor-
mance and success [24, 25, 6]. These works show that even for $O(n)$ non-zeros\textsuperscript{1} in the representations, pursuit methods are expected to succeed with probability one.

2.3  Design of Dictionaries

We now come to the main topic of this thesis, the training of dictionaries based on a set of examples. Given such a set $Y = \{y_i\}_{i=1}^N$, we consider the Sparseland model described in section 1.3 and assume that there exists a dictionary $D$ that gave rise to the given signal examples via sparse combinations, i.e., we assume that there exists $D$ so that solving $(P_0)$ for each example $y_k$ gives a sparse representation $x_k$. It is in this setting that we ask what the proper dictionary $D$ is.

2.3.1  Generalizing the K-Means?

There is an intriguing relation between sparse representation and clustering (i.e., vector quantization). This connection has previously been mentioned in several reports [40, 65, 106]. In clustering, a set of descriptive vectors $\{d_k\}_{k=1}^K$ is learned, and each sample is represented by one of those vectors (the one closest to it, usually in the $\ell^2$ distance measure). We may think of this as an extreme sparse representation, where only one atom is allowed in the signal decomposition, and furthermore, the coefficient multiplying it must be 1. There is a variant of the vector quantization (VQ) coding method, called Gain-Shape VQ, where this coefficient is allowed to vary [49]. In contrast, in sparse representations as discussed in this paper, each example is represented as a linear combination of

\textsuperscript{1} and a proper, somewhat small, constant.
several vectors \(\{\mathbf{d}_k\}_{k=1}^K\). Thus, sparse representations can be referred to as a generalization of the clustering problem.

Since the K-Means algorithm (also known as the generalized Lloyd algorithm - GLA [49]) is the most commonly used procedure for training in the vector quantization setting, it is natural to consider generalizations of this algorithm when turning to the problem of dictionary training. The clustering problem and its K-Means solution will be discussed in more detail in section 4.1, since our work approaches the dictionary training problem by generalizing the K-Means. Here we shall briefly mention that the K-Means process applies two steps per each iteration: (i) given \(\{\mathbf{d}_k\}_{k=1}^K\), assign the training examples to their nearest neighbor; and (ii) given that assignment, update \(\{\mathbf{d}_k\}_{k=1}^K\) to better fit the examples.

The approaches to dictionary design that have been tried so far are very much in line with the two-step process described above. The first step finds the coefficients given the dictionary – a step we shall refer to as “sparse coding”. Then, the dictionary is updated assuming known and fixed coefficients. The differences between the various algorithms that have been proposed are in the method used for the calculation of coefficients, and in the procedure used for modifying the dictionary.

2.3.2 Maximum Likelihood Methods

The methods reported in [71, 82, 81, 72] use probabilistic reasoning in the construction of \(\mathbf{D}\). The proposed model suggests that for every example \(\mathbf{y}\) the relation

\[ \mathbf{y} = \mathbf{D}\mathbf{x} + \mathbf{v}, \]  

(2.15)
holds true with a sparse representation $\mathbf{x}$ and Gaussian white residual vector $\mathbf{v}$ with variance $\sigma^2$. Given the examples $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^N$ these works consider the likelihood function $P(\mathbf{Y}|\mathbf{D})$ and seek the dictionary that maximizes it. Two assumptions are required in order to proceed - the first is that the measurements are drawn independently, readily providing

$$P(\mathbf{Y}|\mathbf{D}) = \prod_{i=1}^N P(\mathbf{y}_i|\mathbf{D}).$$

(2.16)

The second assumption is critical and refers to the “hidden variable” $\mathbf{x}$. The ingredients of the likelihood function are computed using the relation

$$P(\mathbf{y}_i|\mathbf{D}) = \int P(\mathbf{y}_i, \mathbf{x}|\mathbf{D})d\mathbf{x} = \int P(\mathbf{y}_i|\mathbf{x}, \mathbf{D}) \cdot P(\mathbf{x})d\mathbf{x}. \quad (2.17)$$

Returning to the initial assumption in (2.15), we have

$$P(\mathbf{y}_i|\mathbf{x}, \mathbf{D}) = Const \cdot \exp \left\{ \frac{1}{2\sigma^2} \| \mathbf{D}\mathbf{x} - \mathbf{y}_i \|^2 \right\}. \quad (2.18)$$

The prior distribution of the representation vector $\mathbf{x}$ is assumed to be such that the entries of $\mathbf{x}$ are zero-mean iid, with Cauchy [81] or Laplace distributions [71, 82]. Assuming for example a Laplace distribution we get

$$P(\mathbf{y}_i|\mathbf{x}, \mathbf{D}) = \int P(\mathbf{y}_i|\mathbf{x}, \mathbf{D}) \cdot P(\mathbf{x})d\mathbf{x} \\
= Const \cdot \int \exp \left\{ \frac{1}{2\sigma^2} \| \mathbf{D}\mathbf{x} - \mathbf{y}_i \|^2 \right\} \cdot \exp \{ \lambda \|\mathbf{x}\|_1 \} d\mathbf{x}. \quad (2.19)$$

This integration over $\mathbf{x}$ is difficult to evaluate, and indeed, Olshausen and Field [82] handled this by replacing it with the extremal value of $P(\mathbf{y}_i, \mathbf{x}|\mathbf{D})$. The overall problem turns into

$$\mathbf{D} = \arg \max_{\mathbf{D}} \prod_{i=1}^N \max_{\mathbf{x}_i} \{P(\mathbf{y}_i, \mathbf{x}_i|\mathbf{D})\} \quad (2.20)$$

$$= \arg \min_{\mathbf{D}} \sum_{i=1}^N \min_{\mathbf{x}_i} \{\| \mathbf{D}\mathbf{x}_i - \mathbf{y}_i \|^2 + \lambda \|\mathbf{x}_i\|_1 \}.$$
This problem does not penalize the entries of $D$ as it does for the ones of $x_i$. Thus, the solution will tend to increase the dictionary entries’ values, in order to allow the coefficients to become closer to zero. This difficulty has been handled by constraining the $\ell^2$-norm of each basis element, so that the output variance of the coefficients is kept at an appropriate level [81].

An iterative method was suggested for solving (2.20). It includes two main steps in each iteration: (i) calculate the coefficients $x_i$ using a simple gradient descent procedure; and then (ii) update the dictionary using [81]

$$
D^{(n+1)} = D^{(n)} - \eta \sum_{i=1}^{N} (D^{(n)} x_i - y_i)x_i^T.
$$

(2.21)

This idea of iterative refinement, mentioned before as a generalization of the K-Means algorithm, was later used again by other researchers, with some variations [40, 65, 39, 41, 80].

A different approach to handle the integration in (2.19) was suggested by Lewicki and Sejnowski [72]. They approximated the posterior as a Gaussian, enabling an analytic solution of the integration. This allows an objective comparison of different image models (basis or priors). It also removes the need for the additional re-scaling that enforces the norm constraint. However, this model may be too limited in describing the true behaviors expected. This technique and closely related ones have been referred to as approximated ML techniques [65].

There is an interesting relation between the above method and the Independent Component Analysis (ICA) algorithm [3]. The latter handles the case of a complete dictionary (the number of elements equals the dimensionality) without assuming additive noise. The above method is then similar to ICA in that the algorithm can be interpreted as trying to maximize the mutual information between the inputs (samples) and the outputs (the coefficients) [81, 71, 72].
2.3.3 The MOD Method

An appealing dictionary training algorithm, named Method of Optimal Directions (MOD), is presented by Engan et. al. [40, 39, 41]. This method follows more closely the K-Means outline, with a sparse coding stage that uses either OMP or FOCUSS followed by an update of the dictionary. The main contribution of the MOD method is its simple and efficient way of updating the dictionary. Assuming that the sparse coding for each example is known, we define the errors $e_i = y_i - Dx_i$. The overall representation mean square error is given by

$$\|E\|_F^2 = \|[e_1, e_2, \ldots, e_N]\|_F^2 = \|Y - DX\|_F^2.$$  (2.22)

Here we have concatenated all the examples $y_i$ as columns of the matrix $Y$, and similarly gathered the representations coefficient vectors $x_i$ to build the matrix $X$. The notation $\|A\|_F$ stands for the Frobenius Norm, defined as $\|A\|_F = \sqrt{\sum_{ij} A_{ij}^2}$.

Assuming that $X$ is fixed, we can seek an update to $D$ such that the above error is minimized. Taking the derivative of (2.22) with respect to $D$ we obtain the relation $(Y - DX)X^T = 0$, leading to

$$D^{(n+1)} = YX^{(n)T} \cdot (X^{(n)}X^{(n)T})^{-1}$$  (2.23)

MOD is closely related to the work by Olshausen and Field, with improvements both in the sparse coding and the dictionary update stages. Whereas the work in [82, 81, 71] applies a steepest descent to evaluate $x_i$, those are evaluated much more efficiently with either OMP or FOCUSS. Similarly, in updating the dictionary, the update relation given in (2.23) is the best that can be achieved for fixed $X$. The iterative steepest descent update in (2.21) is far slower. Interestingly, in both stages of the algorithm, the difference is in deploying a second order
(Newtonian) update instead of a first-order one. Looking closely at the update relation in (2.21), it could be written as

\[
D^{(n+1)} = D^{(n)} + \eta EX^{(n)T}
\]

(2.24)

\[
= D^{(n)} + \eta(Y - D^{(n)}X^{(n)T})X^{(n)T} = D^{(n)}(I - \eta X^{(n)}X^{(n)T}) + \eta YX^{(n)T}.
\]

Using infinitely many iterations of this sort, and using small enough \(\eta\), this leads to a steady state outcome, which is exactly the MOD update matrix (2.23). Thus, while the MOD method assumes known coefficients at each iteration, and derives the best possible dictionary, the ML method by Olshausen and Field only gets closer to this best current solution, and then turns to calculate the coefficients. Note, however, that in both methods a normalization of the dictionary columns is required and done.

### 2.3.4 Maximum A-posteriori Probability Approach

The same researchers that conceived the MOD method also suggested a maximum a-posteriori probability (MAP) setting for the training of dictionaries, attempting to merge the efficiency of the MOD with a natural way to take into account preferences in the recovered dictionary. In [65, 41, 80, 66] a probablistic point of view is adopted, very similar to the ML methods discussed above. However, rather than working with the likelihood function \(P(Y|D)\), the posterior \(P(D|Y)\) is used. Using Bayes rule, we have \(P(D|Y) \propto P(Y|D)P(D)\), and thus we can use the likelihood expression as before, and add a prior on the dictionary as a new ingredient.

These works considered several priors \(P(D)\) and proposed corresponding formulas for the dictionary update stage. The efficiency of the MOD in these methods is manifested in the efficient sparse coding, which is carried out with
FOCUSS. The proposed algorithms in this family deliberately avoid a direct minimization with respect to $D$ as in MOD, due to the prohibitive $n \times n$ matrix inversion required. Instead, iterative gradient descent is used.

When no prior is chosen, the update formula is the very one used by Olshausen and Field, as in (2.21). A prior that constrains $D$ to have a unit Frobenius norm leads to the update formula

$$D^{(n+1)} = D^{(n)} + \eta E X^T + \eta \cdot \text{tr} \left( X E^T D^{(n)} \right) D^{(n)}.$$  \hfill (2.25)

As can be seen, the first two terms are the same ones as in (2.21). The last term compensates for deviations from the constraint. This case allows different columns in $D$ to have different norm values. As a consequence, columns with small norm values tend to be under-used, as the coefficients they need are larger and as such more penalized.

This shortcoming led to the second prior choice, constraining the columns of $D$ to have a unit $\ell^2$-norm. The new update equation formed is given by

$$d_{i}^{(n+1)} = d_{i}^{(n)} + \eta \left( I - d_{i}^{(n)} d_{i}^{(n)T} \right) E \cdot x_{i}^T,$$  \hfill (2.26)

where $x_{i}^T$ is the $i$-th column in the matrix $X^T$.

Compared to the MOD, this line of work provides slower training algorithms. Simulations reported in [65, 41, 80, 66] on synthetic and real image data seem to provide encouraging results.

### 2.3.5 Unions of Orthonormal Bases

The very recent work reported in [70] considers a dictionary composed as a union of orthonormal bases

$$D = [D_1, \ D_2, \ \ldots, \ D_L],$$
where $D_j \in \mathbb{R}^{n \times n}$, $j = 1, 2, \ldots, L$ are orthonormal matrices. Such a dictionary structure is quite restrictive, but its updating may potentially be made more efficient.

The coefficients of the sparse representations $X$ can be decomposed to $L$ pieces, each referring to a different orthonormal-basis. Thus,

$$X = [X_1^T, X_2^T, \ldots, X_L^T]^T,$$

where $X_i$ is the matrix containing the coefficients of the orthonormal dictionary $D_i$.

One of the major advantages of the union of orthonormal-bases is the relative simplicity of the pursuit algorithm needed for the sparse coding stage. The coefficients are found using the Block Coordinate Relaxation (BCR) algorithm [93]. This is an appealing way to solve $(P_{1,e})$ as a sequence of simple shrinkage steps, such that at each stage $X_i$ is computed, while keeping all the other pieces of $X$ fixed.

Assuming known coefficients, the proposed algorithm updates each orthonormal basis $D_j$ sequentially. The update of $D_j$ is done by first computing the residual matrix

$$E_j = [e_1, e_2, \ldots, e_N] = Y - \sum_{i \neq j} D_i X_i.$$

Then, by computing the singular value decomposition of the matrix $E_j X_j^T = U \Lambda V^T$, the update of the $j$-th orthonormal-basis is done by $D_j = U \Lambda V^T$. This update rule is obtained by solving a constrained least squares problem with $\|E_j - D_j X_j\|^2_F$ as the penalty term, assuming fixed coefficients $X_i$ and error $E_j$. The constraint is over the feasible matrices $D_j$, which are forced to be orthonormal. This way the proposed algorithm improves each matrix $D_j$ separately, by replacing
the role of the data matrix $Y$ in the residual matrix $E_j$, as the latter should be represented by this updated basis.

Compared to previously mentioned training algorithms, the work reported in [70] is different in two important ways; beyond the evident difference of using a structured dictionary rather than a free one, a second major difference is in the proposed sequential update of the dictionary. This update algorithm is reminiscent of the updates done in the K-means. Interestingly, experimental results reported in [70] show weak performance compared to previous methods. This might be explained by the unfavorable coupling of the dictionary parts and their corresponding coefficients, which is overlooked in the update.

2.3.6 Summary of the Prior Art

Almost all previous methods can essentially be interpreted as generalizations of the K-Means algorithm, and yet, there are marked differences between these procedures. In the quest for a successful dictionary training algorithm, there are several desirable properties:

- **Flexibility**: The algorithm should be able to run with any pursuit algorithm, and this way enable choosing the one adequate for the run-time constraints, or the one planned for future usage in conjunction with the obtained dictionary. Methods that decouple the sparse-coding stage from the dictionary update readily have such a property. Such is the case with the MOD and the MAP based methods.

- **Simplicity**: Much of the appeal of a proposed dictionary training method has to do with how simple it is, and more specifically, how similar it is to K-Means. We should have an algorithm that may be regarded as a
natural generalization of the K-Means. The algorithm should emulate the ease with which the K-Means is explainable and implementable. Again, the MOD seems to have made a substantial progress in this direction, although, as we shall see, there is still room for improvement.

- Efficiency: The proposed algorithm should be numerically efficient and exhibit fast convergence. The above described methods are all quite slow. The MOD, which has a second-order update formula, is nearly impractical for very large number of dictionary columns, because of the matrix inversion step involved. Also, in all the above formulations, the dictionary columns are updated before turning to re-evaluate the coefficients. As we shall see later, this approach inflicts a severe limitation on the training speed.

- Well Defined Objective: For a method to succeed, it should have a well defined objective function that measures the quality of the solution obtained. This almost trivial fact was overlooked in some of the preceding work in this field. Hence, even though an algorithm can be designed to greedily improve the representation MSE and the sparsity, it may happen that the algorithm leads to aimless oscillations in terms of a global objective measure of quality.
Chapter 3

Uniqueness of the Dictionary

3.1 Posing the Problem

We are given a set of signals, each of which is known to be a sparse linear combination of atoms from some dictionary, and consider the dictionary search problem, namely, the extraction of a dictionary that can exactly represent each signal sparsely. Formally, the signals are arranged as columns of the matrix $Y \in \mathbb{R}^{n \times N}$ ($n \ll N$), such that $Y = DX$ where $D \in \mathbb{R}^{n \times K}$, and each column in $X$ obeys some known sparsity constraint. Our focus in this section is on whether such a dictionary is unique, and more generally, whether the factorization of the matrix $Y$ under these known sparsity constraints is unique. We did not find any explicit reference to this question, and yet, such uniqueness is implicitly assumed in previous works (e.g., the synthetic experiments in [88, 1] as also in Section 4.2.5).

3.2 Statement of the Uniqueness Result

We now pose a set of assumptions that are mandatory for the uniqueness result we are about to prove. We assume the following:
(1) **Support:** The support of all representation vectors in $\mathbf{X}$ (its columns) satisfy

$$\forall 1 \leq i \leq N, \quad \|\mathbf{x}_i\|_0 = L < \frac{\sigma\{\mathbf{D}\}}{2},$$

where $\sigma\{\mathbf{D}\}$ is the spark of $\mathbf{D}$, defined before in Section 2.2.1. Furthermore, we assume that $L$ is known. Both the knowledge of $L$, and the fact that the representations are assumed to have exactly $L$ non-zeros (and not less) can be easily relaxed. Still, these assumptions are posed for the sake of simplicity of the proof.

(2) **Richness:** The set of examples in $\mathbf{Y}$ includes at least $L + 1$ signals for every possible combination of $L$ atoms in $\mathbf{D}$. Thus, we assume that $\mathbf{Y}$ includes at least $(L+1)^K$ signals. This assumption will later be relaxed, and again, it is posed mainly for simplifying the later analysis.

(3) **Non-Degeneracy:** Given a set of $L + 1$ signals that are built of the same $L$ atoms, their rank is expected to be $L$ or less. We assume that any such set leads to a rank $L$ and not less. Similarly, for any set of $L + 1$ signals which are not generated by the same $L$ atoms, we assume that the rank of such set is necessarily $L + 1$. Both these assumptions mean that no degeneracies in the construction of the signals are allowed. These requirements, although seemingly complicated, only reflect the fact that there are no degenerate ‘coincidences’ in the signals construction. Analogously, considering the signals to be generated with $L$ randomly chosen coefficients in $\mathbf{X}$, these degeneracies are of zero probability.

Based on these assumptions we have the following result:
**Theorem 1:** Under the above assumptions, the factorization of $Y$ is unique, i.e., the factorization $Y = DX$ for which (i) $D \in \mathbb{R}^{n \times K}$ with normalized atoms; and (ii) $X \in \mathbb{R}^{K \times N}$ with $L$ non-zeros in each column, is unique. This uniqueness is up to a right-multiplication of $D$ by a signed permutation matrix, which does not change the desired properties of $D$ and $X$.

### 3.3 The Proof

The proof we provide is constructive (although far from being a practical method to deploy in practice), leading to a pair $\hat{D}$ and $\hat{X}$. Clearly, it is sufficient to prove an equivalence between $\hat{D}$ and $D$ (up to a signed permutation) to guarantee a unique factorization. Given the vector $y_i$ and a dictionary $D$, our assumptions imply that a solution to $y_i = Dx_i$ exists with exactly $L$ non-zeros. Since $L < \frac{\sigma(D)}{2}$, this is the sparsest possible solution and as such it is unique, due to Theorem 1 in 2.2.1. Thus, having found $D$, solving a set of $(P_0)$ problems we necessarily recover the original $X$. Permutations and sign changes do not impact this property, and only change the locations and signs of the entries in $X$ to match the columns in $D$. Thus, it is sufficient to consider the relation between the original dictionary $D$ and the recovered one, $\hat{D}$.

Next, we describe a coherent process with three stages that leads to $\hat{D}$, and show that it necessarily matches the original $D$. The basic steps of this process are (i) Divide the columns of $Y$ into $J = \binom{K}{L}$ sets – $\{G_1, G_2, \ldots, G_J\}$ – each includes all the signals that share the same support (i.e., use the same $L$ atoms from $D$, denoted as $\Omega_j$ for $j = 1, 2, \ldots, J$); (ii) Detect pairs of sets $G_i$ and $G_j$ that share exactly one mutual atom; and (iii) Extract this mutual atom and form $\hat{D}$, which necessarily matches the original $D$. Furthermore, we will expand on each of these
steps.

3.3.1 Stage 1: Clustering The Signals

Due to our earlier support assumption, $L < \sigma\{D\}/2$, and the definition of the spark, every set of $L$ atoms from $D$ is necessarily linearly independent and as such spans an $L$-dimensional subspace. In this stage we identify those $\binom{K}{L}$ subspaces, and divide the signals (columns) in $Y$ according to their embedding subspaces.

The clustering of the columns of $Y$ can be done by first testing the rank of all $\binom{K}{L+1} = (K - L) \cdot J/(L + 1) > J$ sets of $(L + 1)$-tuples from $Y$. If the rank of such a set is $L + 1$, it implies that this set of signals do not belong to the same subspace, and as such it is discarded from further consideration. If the rank equals $L$, it means we have found $L + 1$ signals that belong to one of the subspaces related to a set of atoms $S_j$. The richness assumption assures that such $L + 1$ signals exist per each of the $J$ subspaces, and the non-degeneracy assumption prevents a set of $L + 1$ signals that do not belong to the same subspace to give a rank $L$, and pose as a feasible subspace. Thus, we expect to detect exactly $J$ such successful sets of signals from $Y$, and those, denoted by $\{G_j\}_{j=1}^J$, will serve as the seeds for the overall clustering. Note that the non-degeneracy assumption also implies that the rank of an arbitrary set of $L + 1$ columns cannot be smaller than $L$.

Having found the seed for each of the $J$ sets, $\{G_j\}_{j=1}^J$, we now sweep through all the columns in $Y$ that are not assigned yet, and combine them to one of these $J$ sets, testing again the rank. Due to the non-degeneracy assumption, only one set will give a rank $L$, implying that the tested column belongs to this subspace. All other tests necessarily lead to a rank $L + 1$. 
Using the above procedure, we divide all signals in $Y$ into $\binom{K}{L}$ sets, each of which includes signals that are generated by the same set of $L$ atoms in the original dictionary $D$. The support and richness assumptions ensures that each such subspace will eventually be identified (when testing its corresponding $L+1$ signals). The non-degeneracy assumption ensures that no $L+1$ signals will be mapped into the same set if they were not initially generated by the same $L$ atoms.

### 3.3.2 Stage 2: Detecting Pairs With Mutual Atom

Given the $J$ sets of signals $\{G_j\}_{j=1}^{J}$, we now test the rank of all $J(J-1)/2$ merged pairs (order plays no role). Every two such merged sets, $G_{j_1}$ and $G_{j_2}$, are leaning on two sets of atoms from $D$, $\Omega_{j_1}$ and $\Omega_{j_2}$. If the intersection between these two sets is empty, $|\Omega_{j_1} \cap \Omega_{j_2}| = 0$, then the rank is necessarily $2L$. It cannot be higher as each of the sets have a rank $L$, and cannot be smaller due to the non-degeneracy assumption and the fact that $2L < \sigma(D)$. If the intersection includes one atom, $|\Omega_{j_1} \cap \Omega_{j_2}| = 1$, the rank is necessarily $2L-1$ for the same reasons. Getting a rank smaller than $2L-1$ means a larger intersection. Thus, we take only those pairs that lead to rank $2L-1$.

Let us consider the first atom as the single interaction between pairs $\{G_{j_1}, G_{j_2}\}$. How many such pairs will be found? Putting this atom aside, we remain with $K-1$ atoms from which we have to choose $2L-2$ atoms to participate in the construction of the two sets. These should be divided into two sets of $L-1$ atoms each, and only half count (order, as before, is redundant). Therefore, we have $M = 0.5 \cdot \binom{K-1}{2L-2} \binom{2L-2}{L-1}$ such pairs that intersect only on the first atom. Thus, every atom in the original dictionary $D$ can and will be found many times, calling for a pruning. Both the evaluation of this atom from the intersection and the
pruning are parts of the next and last stage.

### 3.3.3 Stage 3: Extracting the Mutual Atom

Assume that we have the pair of sets \( \{ G_{j_1}, G_{j_2} \} \) known to intersect on one atom. Taking an arbitrary \( L \) signals from \( G_{j_1} \), they span the same \( L \)-dimensional subspace as the atoms in \( \Omega_{j_1} \). Thus, gathering these \( L \) signals into a matrix \( Y_{j_1} \in \mathcal{R}^{n \times L} \), there exists a vector \( v_1 \in \mathcal{R}^{L \times 1} \) such that \( Y_{j_1}v_1 \) is parallel to the desired intersection atom. Similarly, taking \( L \) arbitrary members from \( G_{j_2} \) and forming the matrix \( Y_{j_2} \in \mathcal{R}^{n \times L} \), there exists a vector \( v_2 \in \mathcal{R}^{L \times 1} \) such that \( Y_{j_2}v_2 \) is parallel to the same intersection atom. Thus, we have the relationship

\[
Y_{j_1}v_1 = Y_{j_2}v_2, \tag{3.2}
\]

or, posed differently, this relationship leads to the homogeneous linear system of equations

\[
[Y_{j_1}, -Y_{j_2}]v = 0, \tag{3.3}
\]

where \( v \) is a vertical concatenation of \( v_1 \) and \( v_2 \). The constructed matrix has \( n \) rows and \( 2L \) columns, but we already know that its rank is \( 2L - 1 \) due to the intersection. A vector \( v \) in its null-space leads to the desired \( v_1 \) and \( v_2 \), and they can be obtained as the last right singular vector in an SVD operation [51]. Having found \( v_1 \), the term \( Y_{j_1}v_1 \) stands for the desired intersection atom, being parallel to a true atom found in \( D \) up to a scalar multiplication.

Repeating the above process for each pair with a single atom intersection, we obtain \( M \cdot K \) candidate atoms. Starting with the first, we sweep through this set and seek all others that are parallel to it, pruning them. This process proceeds
for the second, third, and till we remain with only $K$ atoms. These are the desired atoms, being the columns of the original dictionary $D$.

### 3.3.4 Summarizing the Proof

Up until now we have presented a constructive algorithm for the extraction of the dictionary $D$ that was used in the construction of $Y$. Indeed, in the algorithm described there are multiple possibilities of choosing the pairs in the second stage, as well as choosing the $L$ elements that construct the matrices $Y_{j_1}$ and $Y_{j_2}$ in the third stage. Nevertheless, all possible choices lead to the same solution $D$ up to simple transformations. Thus, the matrix $Y$, created as the product $DX$, is factorized as desired.

Could there be a different feasible factorization? Let us assume that there exists such a second different factorization $Y = \tilde{D}\tilde{X}$. Executing the above algorithm on $Y$ must lead to the matrix $\tilde{D}$, due to the constructive method we have developed. On the other hand, this algorithm must also lead to $D$ for the same reasons. Thus, we necessarily conclude that $\tilde{D}$ must be equivalent to $D$, and therefore the factorization is unique.  

### 3.4 Summary

While correct, the above uniqueness proof is totally impractical. The daunting number of signals required, as well as the computational cost, prevent this algorithm from being practical. However, keeping this method in mind, the requirement on the number of signals can be relaxed, leading to a reduced computational complexity. We have seen that in the algorithm developed there are severe redundancies in building the atoms. How much lower could the number
of examples go and still lead to a successful factorization? As an example that illustrates the possibilities, we could take exactly $L + 1$ examples per each set, but consider only $2K$ such sets. If those sets are chosen smartly to divide into pairs that overlap on each of the $K$ atoms, this would be sufficient for the success of the algorithm. Thus, $2K(L + 1)$ signal examples could in principle be used. In fact, even such a set could be found redundant, due to the ability to cross pairs differently and exploit other overlaps.

However, this relaxation still prevents this method from being applied in practice, as the algorithm still requires a sweep through all combinations of $L$ signals out of the whole set. Yet, it raises the theoretical question concerning the minimal number of signals that can guarantee uniqueness of the dictionary under sparsity constraints – a question we leave open at this stage.

Up until now, we discussed the case of an exact representation of the signals. This constraint, by itself, is not expected to be held in practice, where some representation error is always allowed. In such a case, the above proof, even under the most restrictive conditions, does not hold. We believe uniqueness in the case of an approximated representation no longer exists, but stability does. That is, the smaller the allowed representation error, and the larger the set of given signals, the more we can expect the extracted dictionary to resemble the original one. Yet, such a proof is not trivial and left here for future work.
Chapter 4

The K-SVD Algorithm

In this section we introduce the K-SVD algorithm for training of dictionaries. This algorithm is flexible and works in conjunction with any pursuit algorithm. It is simple and designed to be a truly direct generalization of the K-Means. As such, when forced to work with one atom per signal, it trains a dictionary for the Gain-Shape VQ. When forced to have a unit coefficient for this atom, it exactly reproduces the K-Means algorithm. The K-SVD is highly efficient, due to an effective sparse coding, and a Gauss-Seidel-like accelerated dictionary update method. The algorithm’s steps are coherent with each other, both working towards the minimization of a clear overall objective function.

4.1 K-Means Algorithm for Vector Quantization

We start our discussion with a description of the K-Means, setting the notation for the rest of this section. While this may seem superfluous, we will use the very description of the K-Means to derive the K-SVD as its direct extension. We then discuss some of the K-SVD properties and implementation issues.

A codebook that includes $K$ codewords (representatives) is used to represent a wide family of vectors (signals) $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^N$ ($N \gg K$) by nearest neighbor
assignment. This leads to an efficient compression or description of those signals, as clusters in $\mathcal{R}^n$ surrounding the chosen codewords. As a side note we remind the reader that based on the expectation maximization procedure, the K-Means can be extended to suggest a fuzzy assignment and a covariance matrix per each cluster, so that the data is modeled as a mixture of Gaussians [17].

The dictionary of VQ codewords is typically trained using the K-Means algorithm, and as we have argued before, this has a close resemblance to the problem studied in this thesis. We denote the codebook matrix by $C = [c_1, c_2, \ldots, c_K]$, the codewords being the columns. When $C$ is given, each signal is represented as its closest codeword (under $\ell^2$-norm distance). We can write $y_i = Cx_i$, where $x_i = e_j$ is a vector from the trivial basis, with all zero entries except a one in the $j$-th position. The index $j$ is selected such that

$$\forall \ k \neq j \ \| y_i - Ce_j \|_2^2 \leq \| y_i - Ce_k \|_2^2.$$  

(4.1)

This is considered as an extreme case of sparse coding in the sense that only one atom is allowed to participate in the construction of $y_i$, and the coefficient is forced to be 1. The representation MSE per $y_i$ is defined as $e_i^2 = \| y_i - Cx_i \|_2^2$, and the overall MSE is

$$E = \sum_{i=1}^{K} e_i^2 = \| Y - CX \|_F^2.$$  

(4.2)

The VQ training problem is to find a codebook $C$ that minimizes the error $E$, subject to the limited structure of $X$, whose columns must be taken from the trivial basis,

$$\min_{C,X} \{ \| Y - CX \|_F^2 \} \ \text{subject to} \ \forall i, \ x_i = e_k \ \text{for some} \ k.$$  

(4.3)

The K-Means algorithm is an iterative method used for designing the optimal codebook for VQ [49]. In each iteration there are two stages - one for sparse
coding that essentially evaluates $X$, and one for updating the codebook. Figure 4.1 gives a more detailed description of these steps.

The sparse coding stage assumes a known codebook $C^{(J-1)}$, and computes a feasible $X$ that minimizes the value of (4.3). Similarly, the dictionary update stage fixes $X$ as known, and seeks an update of $C$ so as to minimize (4.3). Clearly, at each iteration either a reduction or no change in the MSE is ensured. Furthermore, at each such stage, the minimization step is optimal under the assumptions. Note that we have deliberately chosen not to discuss stopping rules for the above-described algorithm, since those vary a lot but are quite easy to handle [49].

4.2 The K-SVD Algorithm

4.2.1 K-SVD – Generalizing the K-Means

The sparse representation problem can be viewed as a generalization of the VQ objective (4.3), in which we allow each input signal to be represented by a linear combination of codewords, which we now call dictionary elements. Therefore the coefficients vector is now allowed more than one nonzero entry, and these can have arbitrary values. For this case, the minimization corresponding to Equation (4.3) generalizes to the search of the best possible dictionary for the sparse representation of the example set $Y$, namely,

$$\min_{D,X} \{ \| Y - DX \|_F^2 \} \quad \text{subject to} \quad \forall i, \| x_i \|_0 \leq T_0. \quad (4.4)$$

A similar objective could be posed, considering

$$\min_{D,X} \sum_i \| x_i \|_0 \quad \text{subject to} \quad \| Y - DX \|_F^2 \leq \epsilon, \quad (4.5)$$

for a fixed value $\epsilon$. In this section we mainly discuss the first problem (4.4), although the treatment of the second formulation is very similar.
Task: Find the best possible codebook to represent the data samples \( \{y_i\}_{i=1}^N \) by nearest neighbor, by solving

\[
\min_{C,X} \left\{ \|Y - CX\|_F^2 \right\} \quad \text{subject to} \quad \forall i, x_i = e_k \text{ for some } k.
\]

Initialization: Set the codebook matrix \( C^{(0)} \in \mathcal{R}^{n \times K} \). Set \( J = 1 \).

Repeat until convergence (use stop rule):

- **Sparse Coding Stage:** Partition the training samples \( Y \) into \( K \) sets

\[
(R_1^{(J-1)}, R_2^{(J-1)}, \ldots, R_K^{(J-1)}),
\]

each holding the sample indices most similar to the column \( c_k^{(J-1)} \),

\[
R_k^{(J-1)} = \left\{ i \mid \forall t \neq k, \|y_i - c_k^{(J-1)}\|_2 < \|y_i - c_t^{(J-1)}\|_2 \right\}.
\]

- **Codebook Update Stage:** For each column \( k \) in \( C^{(J-1)} \), update it by

\[
c_k^{(J)} = \frac{1}{|R_k|} \sum_{i \in R_k^{(J-1)}} y_i.
\]

- Set \( J = J + 1 \).

Figure 4.1: The K-Means Algorithm
In our algorithm we minimize the expression in (4.4) iteratively. First, we fix $\mathbf{D}$ and aim to find the best coefficient matrix $\mathbf{X}$ that can be found. As finding the truly optimal $\mathbf{X}$ is impossible, we use an approximation pursuit method. Any such algorithm can be used for the calculation of the coefficients, as long as it can supply a solution with a fixed and predetermined number of nonzero entries, $T_0$.

Once the sparse coding task is done, a second stage is performed to search for a better dictionary. This process updates one column at a time, fixing all columns in $\mathbf{D}$ except one, $\mathbf{d}_k$, and finding a new column $\tilde{\mathbf{d}}_k$ and new values for its coefficients that best reduce the MSE. This is markedly different from all the K-Means generalizations that were described in Section 2.3. All those methods freeze $\mathbf{X}$ while finding a better $\mathbf{D}$. Our approach is different, as we change the columns of $\mathbf{D}$ sequentially, and allow changing the relevant coefficients as well. In a sense, this approach is a more direct generalization of the K-Means algorithm, because it updates each column separately, as done in K-Means. One may argue that in K-Means the nonzero entries in $\mathbf{X}$ are fixed during the improvement of $\mathbf{c}_k$, but as we shall see next, this is true because in the K-Means (and the gain-shape VQ), the column update problems are decoupled, whereas in the more general setting this need not be the case.

The process of updating only one column of $\mathbf{D}$ at a time is a problem having a straightforward solution based on the singular value decomposition (SVD). Furthermore, allowing a change in the coefficient values while updating the dictionary columns accelerates convergence, since the subsequent column updates will be based on more relevant coefficients. The overall effect is very much in line with the leap from gradient descent to Gauss-Seidel methods in optimization.

Here one might be tempted to suggest skipping the step of sparse coding,
and using only updates of columns in \(D\), along with their coefficients, applied in a cyclic fashion, again and again. This however will not work well, as the support of the representations will never be changed, and such an algorithm will necessarily fall into a low-quality solution.

### 4.2.2 K-SVD - Detailed Description

We shall now discuss the K-SVD in detail. Recall that our objective function is

\[
\min_{D,X} \left\{ \|Y - DX\|_F^2 \right\} \quad \text{subject to } \forall i, \|x_i\|_0 \leq T_0. \tag{4.6}
\]

Let us first consider the sparse coding stage, where we assume that \(D\) is fixed, and consider the above optimization problem as a search for sparse representations with coefficients summarized in the matrix \(X\). The penalty term can be rewritten as

\[
\|Y - DX\|_F^2 = \sum_{i=1}^{N} \|y_i - Dx_i\|_2^2.
\]

Therefore the problem posed in (4.6) can be decoupled to \(N\) distinct problems of the form

\[
i = 1, 2, \ldots, N, \min_{x_i} \left\{ \|y_i - Dx_i\|_2^2 \right\} \quad \text{subject to } \|x_i\|_0 \leq T_0. \tag{4.7}
\]

This problem is adequately addressed by the pursuit algorithms discussed in Section 2.1, and we have seen that if \(T_0\) is small enough, their solution is a good approximation to the ideal one that is numerically infeasible to compute directly.

We now turn to the second, and slightly more involved process of updating the dictionary together with the nonzero coefficients. Assume that both \(X\) and \(D\) are fixed, and we put in question only one column in the dictionary, \(d_k\), and the
coefficients that correspond to it, the $k$-th row in $X$, denoted as $x^k_T$ (this is not the vector $x_k$ which is the $k$-th column in $X$). Returning to the objective function (4.6), the penalty term can be rewritten as

$$\|Y - DX\|_F^2 = \left\| \left( Y - \sum_{j=1}^{K} d_j x^j_T \right) - d_k x^k_T \right\|_F^2 = \left\| E_k - d_k x^k_T \right\|_F^2.$$

We have decomposed the multiplication $DX$ to the sum of $K$ rank-1 matrices. Among those, $K - 1$ terms are assumed fixed, and one – the $k$-th – remains in question. The matrix $E_k$ stands for the error for all the $N$ examples when the $k$-th atom is removed. Note the resemblance between this error and the one defined in [70].

Here, it would be tempting to suggest the use of the SVD to find alternative $d_k$ and $x^k_T$. The SVD finds the closest rank-1 matrix (in Frobenius norm) that approximates $E_k$, and this will effectively minimize the error as defined in (4.8). However, such a step will be a mistake, because the new vector $x^k_T$ is very likely to be filled, since in such an update of $d_k$ we do not enforce the sparsity constraint.

A remedy to the above problem, however, is simple and also quite intuitive. Define $\omega_k$ as the set of indices pointing to examples $\{y_i\}$ that use the atom $d_k$, i.e., those where $x^k_T(i)$ is nonzero. Thus,

$$\omega_k = \{i \mid 1 \leq i \leq K, x^k_T(i) \neq 0\}.$$  

Define $\Omega_k$ as a matrix of size $N \times |\omega_k|$, with ones on the $(\omega_k(i), i)$-th entries, and zeros elsewhere. When multiplying $x^k_R = x^k_T \Omega_k$, this shrinks the row vector $x^k_T$ by discarding of the zero entries, resulting with the row vector $x^k_R$ of length
Similarly, the multiplication $Y_k^R = Y\Omega_k$ creates a matrix of size $n \times |\omega_k|$ that includes a subset of the examples that are currently using the $d_k$ atom. The same effect happens with $E_k^R = E_k\Omega_k$, implying a selection of error columns that correspond to examples that use the atom $d_k$.

With this notation, we may now return to (4.8) and suggest minimization with respect to both $d_k$ and $x_k^R$, but this time force the solution of $x_k^R$ to have the same support as the original $x_k^R$. This is equivalent to the minimization of

$$
\|E_k\Omega_k - d_k x_k^R \Omega_k\|_F^2 = \|E_k^R - d_k x_k^R\|_F^2,
$$

and this time it can be done directly via SVD. In fact, the above simply takes the error as appears in Equation (4.8) and splits it to two additive parts - the first that refers to the examples using the $k$-th atom, and the second that refers to the remaining ones. The above change suggests to minimize only the error for the first set of examples.

Taking the restricted matrix $E_k^R$, an SVD operation decomposes it to $E_k^R = U\Delta V^T$. We define the solution for $\tilde{d}_k$ as the first column of $U$, and the coefficient vector $x_k^R$ as the first column of $V$ multiplied by $\Delta(1, 1)$. Note that in this solution we necessarily have that (i) the columns of $D$ remain normalized; and (ii) the support of all representations either stays the same or gets smaller by possible nulling of terms. Alternatively to the SVD computation, a few iterations of the following procedure, followed by the scaling of the two vectors for normalization of $d_k$ can derive the same solution,

$$
\tilde{d}_k = \frac{E_k^R x_k^R}{x_k^R \cdot x_k^R}, \quad x_k^R = \frac{\tilde{d}_k^T E_k^R}{\tilde{d}_k^T \cdot d_k}.
$$

We call this algorithm the “K-SVD” to parallel the name K-Means. While K-Means applies K computations of mean (averaging) to update the codebook, the
K-SVD obtains the updated dictionary by K SVD computations, each determining one column. A full description of the algorithm is given in Figure 4.2.

In the K-SVD algorithm we sweep through the columns and use always the most updated coefficients as they emerge from preceding SVD steps. Parallel versions of this algorithm can also be considered, where all updates of the previous dictionary are done based on the same $X$. Experiments show that while this version also converges, it yields an inferior solution, and typically requires more than 4 times the number of iterations.

An important question that arises is whether the K-SVD converges. Let us first assume we can perform the sparse coding stage perfectly, retrieving the best approximation to the signal $y_i$ that contains no more than $T_0$ nonzero entries. In this case, and assuming a fixed dictionary $D$, each sparse coding step decreases the total representation error $\|Y - DX\|_F^2$, posed in (4.6). Moreover, at the update step for $d_k$, an additional reduction or no change in the MSE is guaranteed, while not violating the sparsity constraint. Executing a series of such steps ensures a monotonic MSE reduction, and therefore, a convergence of the algorithm.

Unfortunately, the above claim depends on the success of pursuit algorithms to robustly approximate the solution to (4.7), and thus convergence is not always guaranteed. However, when $T_0$ is small enough relative to $n$, the OMP, the FOCUSS, and the BP approximating methods are known to perform very well\footnote{While OMP can be naturally used to get a fixed and pre-determined number of non-zeros ($T_0$), both BP and FOCUSS require some slight modifications. For example, in using FOCUSS to derive exactly $T_0$ non-zero coefficients, the regularization parameter should be adapted while iterating.}. In those circumstances the convergence is guaranteed. We can also ensure convergence by external interference - by comparing the best solution using the already given support to the one proposed by the new run of the pursuit algorithm, and
Task: Find the best dictionary to represent the data samples \( \{ y_i \}_{i=1}^N \) as sparse compositions, by solving
\[
\min_{D, X} \left\{ \| Y - DX \|_F^2 \right\} \quad \text{subject to} \quad \forall i, \| x_i \|_0 \leq T_0.
\]

Initialization: Set the dictionary matrix \( D^{(0)} \in \mathcal{R}^{n \times K} \) with \( \ell^2 \) normalized columns. Set \( J = 1 \).

Repeat until convergence (stopping rule):

- **Sparse Coding Stage:** Use any pursuit algorithm to compute the representation vectors \( x_i \) for each example \( y_i \), by approximating the solution of
  \[
i = 1, 2, \ldots, N, \min_{x_i} \left\{ \| y_i - Dx_i \|_2^2 \right\} \quad \text{subject to} \quad \| x_i \|_0 \leq T_0.
  \]

- **Codebook Update Stage:** For each column \( k = 1, 2, \ldots, K \) in \( D^{(J-1)} \), update it by
  * Define the set of examples that use this atom, \( \omega_k = \{ i \mid 1 \leq i \leq N, x_i^k(i) \neq 0 \} \).
  * Compute the overall representation error matrix, \( E_k \), by
    \[
    E_k = Y - \sum_{j \neq k} d_j x_j^T.
    \]
  * Restrict \( E_k \) by choosing only the columns corresponding to \( \omega_k \), and obtain \( E_k^R \).
  * Apply SVD decomposition \( E_k^R = U \Delta V^T \). Choose the updated dictionary column \( d_k \) to be the first column of \( U \). Update the coefficient vector \( x_k^R \) to be the first column of \( V \) multiplied by \( \Delta(1,1) \).

- Set \( J = J + 1 \).

Figure 4.2: The K-SVD Algorithm.
adopting the better one. This way we shall always get an improvement. Practically, we saw in all our experiments that a convergence is reached, and there was no need for such external interference.

Yet, convergence of the algorithm is not guaranteed to be to a local minimum solution, for which each perturbation of one or more of the variables necessarily derives an increased value of the cost function (Equation 4.6). The K-SVD, just like the K-Means algorithm, is not guaranteed to converge to such a solution, and may well be trapped in a stable point of the numerical algorithm. An example of such a stable point is the following. Lets assume that the dictionary is initialized such that all signals are better represented using only $K - 1$ out of the available $K$ atoms. Such a case definitely results convergence to a sub-optimal solution (the unused atom can be changed to better represent at least one signal without increasing the size of the support). Several steps to avoid such unwanted solutions are described later in Section 4.2.4. Yet still, avoiding them completely is not promised.

To summarize the K-SVD description, we discuss its computational complexity. We should consider both stages - sparse coding and dictionary update. The complexity of the sparse coding stage depends on the selected pursuit method. When using the OMP and neglecting the least-squares steps within, each coefficient for each signal can be found in $O(nK)$ operations, which results with $O(nNLK)$ operations for the whole sparse coding stage. In the dictionary update stage there are $K$ updates. The average number of signals that use each atom is $NL/K$, which results with an error matrix of size $n \times NL/K$ on average. Considering the iterative method for finding the rank-1 solution, this leads to the requirement for $O(nNL/K)$ operations, which results a total of $O(nNL)$ operations for the whole
stage, if implemented smartly (special data structures should be formed in order to retrieve the relevant signals and coefficients without additional costs). When considering the MOD algorithm described in Section 2.3, each dictionary update stage requires an inversion of a $K \times K$ matrix, as also $O(K^2N)$ operations for matrix multiplications, which results a total of $O(K^2N)$ operations.

Combining the two stages, the full K-SVD algorithm requires $O(nNLK)$ operations for each iteration. Clearly, acceleration of the algorithm can be done by replacing the pursuit method, as being the most expensive stage in the algorithm. The number of iterations until convergence depends on the application and on the initial dictionary (see Section 4.2.5).

4.2.3 From K-SVD Back to K-Means

What happens when we force $T_0 = 1$? This case corresponds to the gain-shape VQ, and as such, it is important as the K-SVD becomes a method for its codebook training. When $T_0 = 1$ the coefficient matrix $X$ has only one nonzero entry per column. Thus, computing the error $E_k^R$ in (4.10), yields

$$E_k^R = E_k \Omega_k = \left( Y - \sum_{j \neq k} d_j x_j^T \right) \Omega_k = Y \Omega_k = Y_k^R.$$  (4.12)

This is because the restriction $\Omega_k$ takes only those columns in $E_k$ that use the $d_k$ atom, and thus necessarily, they use no other atoms, implying that for all $j$, $x_j^T \Omega_k = 0$.

The implication of the above outcome is that the SVD in the $T_0 = 1$ case is done directly on the set of examples in $\omega_k$. Also, the $K$ updates of the columns of $D$ become independent of each other, implying that a sequential process as before, or a parallel one, both lead to the same algorithm. We mentioned before that
the K-Means update of the cluster centroids could be interpreted as a sequential process, and the discussion here sheds some further light on this interpretation.

We could further constrain our representation stage and, beyond the choice $T_0 = 1$, limit the nonzero entries of $X$ to be 1. This brings us back to the classical clustering problem as described earlier. In this case we have that $x^k_R$ is filled with ones, thus $x^k_R = 1^T$. The K-SVD then needs to approximate the restricted error matrix $E^R_k = Y^R_k$ by a rank-1 matrix $d_k \cdot 1^T$. The solution is the mean of the columns of $Y^R_k$, exactly as K-Means suggests.

### 4.2.4 K-SVD - Implementation Details

Just like the K-Means, the K-SVD algorithm is susceptible to sub-optimal solutions. Our experiments show that improved results can be reached if the following variations are applied:

- When using approximation methods with a fixed number of coefficients, we found that FOCUSS proves to be the best in terms of getting the best out of each iteration. However, from a run-time point of view, OMP was found to lead to far more efficient overall algorithm.

- When a dictionary element is not being used ‘enough’ (relative to the number of dictionary elements and to the number of samples) it could be replaced with the least-represented signal element, after being normalized (the representation is measured without the dictionary element that is going to be replaced). Since the number of data elements is much larger than the number of dictionary elements, and since our model assumption suggests that the dictionary atoms are of equal importance, such replacement is very effective in avoiding local minima and over-fitting.
• Similar to the idea of removal of unpopular elements from the dictionary, we found that it is very effective to prune the dictionary from having too-close elements. If indeed such a pair of atoms is found (based on their absolute inner product exceeding some threshold), one of those elements should be removed and replaced with the least-represented signal element.

Similarly to the K-Means, we can propose a variety of techniques to further improve the K-SVD algorithm. Most appealing on this list are multiscale approaches, and tree-based training where the number of columns $K$ is allowed to increase during the algorithm. Initial trial of adapting the K-SVD for training multiscale dictionary is presented in Section 6.4.2.

4.2.5 Synthetic Experiments

As in previously reported works [65, 70], we first try the K-SVD algorithm on synthetic signals, to test whether this algorithm recovers the original dictionary that generated the data, and to compare its results with other reported algorithms. The experiment we describe follows the following steps:

• **Generation of the data to train on:** A random matrix $D$ (referred to later-on as the *generating dictionary*) of size $20 \times 50$ was generated with iid uniformly distributed entries. Each column was normalized to a unit $\ell^2$-norm. Then, 1500 data signals $\{y_i\}_{i=1}^{1500}$ of dimension 20 were produced, each created by a linear combination of 3 different generating dictionary atoms, with uniformly distributed iid coefficients in random and independent locations. White Gaussian noise with varying SNR was added to the resulting data signals.
• **Applying the K-SVD:** The dictionary was initialized with 50 data signals. The coefficients were found using OMP with a fixed number of 3 coefficients. The maximum number of iterations was set to 80.

• **Comparison to other reported works:** We implemented the MOD algorithm, and applied it on the same data, using OMP with a fixed number of 3 coefficients, and initializing in the same way. We executed the MOD algorithm for a total number of 80 iterations. We also executed the MAP-based algorithm of Kreutz-Delgado et. al. [65]\(^2\). This algorithm was executed as is, therefore using FOCUSS as its decomposition method. Here, again, a maximum of 80 iterations were allowed.

• **Results:** The computed dictionary was compared against the known generating dictionary. This comparison was done by sweeping through the columns of the generating dictionary, and finding the closest column in the computed dictionary, measuring the distance via

\[
1 - |d_i^T \hat{d}_i|, \tag{4.13}
\]

where \(d_i\) is a generating dictionary atom, and \(\hat{d}_i\) is its corresponding element in the recovered dictionary. A distance less than 0.01 was considered a success. All trials were repeated 50 times, and the number of successes in each trial was computed. Figure 4.3 displays the results for the three algorithms for noise levels of 10dB, 20dB, 30dB and for the noiseless case. We should note that for different dictionary size (e.g., 20 \(\times\) 30) and with more executed iterations, the MAP-based algorithm improves and get closer to the K-SVD detection rates.

\(^2\) The authors of [65] have generously shared their software with us.
Figure 4.3: Synthetic results: for each of the tested algorithms and for each noise level, 50 trials were conducted, and their results sorted. The graph labels represent the mean number of detected atoms (out of 50) over the ordered tests in sets of 10 experiments.
4.3 Applications to Image Processing - Basic Results

We carried out several experiments on natural image data, trying to show the practicality of the proposed algorithm and the general sparse coding theme. We should emphasize that our tests here come only to prove the concept of using such dictionaries with sparse representations. In the next chapter we introduce a large-scale real problem for which we demonstrate the relevance of the K-SVD and its high potential. The experiments we describe follow the following steps:

- **Training Data:** The training data was constructed as a set of 11,000 examples of block patches of size $8 \times 8$ pixels, taken from a database of face images (in various locations). A random collection of 500 such blocks, sorted by their variance, is presented in Figure 4.4.

- **Removal of the DC:** Working with real image data we preferred that all dictionary elements except one have a zero mean. The same measure was practiced in previous work [82]. For this purpose, the mean value of all patches was subtracted off before applying the K-SVD algorithm.

- **Running the K-SVD:** We applied the K-SVD, training a dictionary of size $64 \times 441$ for 80 iterations. The choice $K = 441$ came from our attempt to compare the outcome to the overcomplete Haar dictionary of the same size (see the following section). The coefficients were computed using OMP with a fixed number of 10 coefficients per patch. Note that better performance can be obtained by switching to FOCUSS. We concentrated on OMP because of its simplicity and fast execution. The trained dictionary is presented in Figure 4.5.
• **Comparison Dictionaries**: The trained dictionary was compared with the overcomplete Haar dictionary which includes separable basis functions, having steps of various sizes and in all locations (total of 441 elements). In addition, we build an overcomplete separable version of the DCT dictionary by sampling the cosine wave in different frequencies to result with a total of 441 elements. The overcomplete Haar dictionary and the overcomplete DCT dictionary are presented in Figures 4.6 and 4.7, respectively.

• **Applications**: We used the K-SVD results, denoted here as the learned dictionary, for two different applications on images. All tests were performed on one face image which was not included in the training set. The first application is filling-in missing pixels: we deleted random pixels in the image, and filled their values using the various dictionaries decomposition. We then tested the compression potential of the learned dictionary decomposition, and derived a rate-distortion graph. We hereafter describe those experiments in more detail.

![Figure 4.4: A collection of 500 random blocks that were used for training, sorted by their variance.](image)
Figure 4.5: The learned dictionary. The atoms are sorted in an ascending order of their variance, and stretched to maximal range for display purposes.

Figure 4.6: The overcomplete separable Haar dictionary.
Figure 4.7: The overcomplete DCT dictionary.
4.3.1 Filling-In Missing Pixels

We chose one random full face image, which consists of 594 non-overlapping blocks (none of which were used for training). For each block, the following procedure was conducted for \( r \) in the range \( \{0.2, 0.9\} \):

1. A fraction \( r \) of the pixels in each block, in random locations, was deleted (set to zero).

2. The coefficients of the corrupted block under the learned dictionary, the overcomplete Haar dictionary, and the overcomplete DCT dictionary were found using OMP with an error bound of \( \|0.02 \cdot 1\|_2 \), where \( 1 \in \mathcal{R}^n \) is a vector of all ones\(^3\), (allowing an error of \( \pm 5 \) gray-values in 8-bit images). All projections in the OMP algorithm included only the non-corrupted pixels, and for this purpose, the dictionary elements were normalized so that the non-corrupted indices in each dictionary element have a unit norm. The resulting coefficient vector of the block \( B \) is denoted \( x_B \).

3. The reconstructed block \( \hat{B} \) was chosen as \( \hat{B} = D \cdot x_B \).

4. The reconstruction error was set to: \( \sqrt{\|B - \hat{B}\|_F^2 / 64} \) (64 is the number of pixels in each block).

The mean reconstruction errors (for all blocks and all corruption rates) were computed, and are displayed in Figure 4.8. Two corrupted images and their reconstructions can be seen in Figure 4.9. As can be seen, higher quality recovery is obtained using the learned dictionary.

\(^3\) The input image is scaled to the dynamic range \([0, 1]\).
Figure 4.8: The RMSE for 594 new blocks with missing pixels using the learned dictionary, overcomplete Haar dictionary, and overcomplete DCT dictionary.
Figure 4.9: The corrupted image (left) with the missing pixels marked as points, and the reconstructed results by the learned dictionary, the overcomplete Haar dictionary, and the overcomplete DCT dictionary, respectively. The different rows are for 50% and 70% of missing pixels.
4.3.2 Compression

A compression comparison was conducted between the overcomplete learned dictionary, the overcomplete Haar dictionary, and the overcomplete DCT dictionary (as explained before), all of size 64 × 441. In addition, we compared to the regular (unitary) DCT dictionary (used by the JPEG algorithm). The resulting rate-distortion graph is presented in Figure 4.10. In this compression test, the face image was partitioned (again) into 594 disjoint 8 × 8 blocks. All blocks were coded in various rates (bits-per-pixel values), and the PSNR was measured. Let I be the original image and \( \tilde{I} \) be the coded image, combined by all the coded blocks. We denote \( e^2 \) as the mean squared error between I and \( \tilde{I} \), and

\[
PSNR = 10 \cdot \log_{10} \left( \frac{1}{e^2} \right). \tag{4.14}
\]

In each test we set an error goal \( e \), and fixed the number of bits-per-coefficient \( Q \). For each such pair of parameters, all blocks were coded in order to achieve the desired error goal, and the coefficients were quantized to the desired number of bits (uniform quantization, using upper and lower bounds for each coefficient in each dictionary based on the training set coefficients). For the overcomplete dictionaries, we used the OMP coding method. The rate value was defined as

\[
R = \frac{a \cdot \frac{1}{2} \text{Blocks} + \frac{1}{2} \text{coefs} \cdot (b + Q)}{\frac{1}{2} \text{pixels}}, \tag{4.15}
\]

where

- \( a \) holds the required number of bits to code the number of coefficients for each block.

- \( b \) holds the required number of bits to code the index of the representing atom. Both \( a \) and \( b \) values were calculated using an entropy coder.


- \( \#Blocks \) is the number of blocks in the image (594).

- \( \#coeffs \) is the total number of coefficients required to represent the whole image.

- \( \#pixels \) is the number of pixels in the image (= 64 \( \#Blocks \)).

Figure 4.10: Compression results: Rate-Distortion graphs.

In the unitary DCT dictionary we picked the coefficients in a zig-zag order, as done by JPEG, until the error bound \( e \) is reached. Therefore, the index of each atom should not be coded, and the rate was defined by,

\[
R = \frac{a \cdot \#Blocks + \#coeffs \cdot Q}{\#pixels},
\]

(4.16)

with the same notation as before.
By sweeping through various values of $e$ and $Q$ we get per each dictionary several curves in the R-D plane. Figure 4.10 presents the best obtained R-D curves for each dictionary. As can be seen, the K-SVD dictionary outperforms all other dictionaries, and achieves up to $1 - 2dB$ better for bit rates less than 1.5 bits-per-pixel (where the sparsity model holds true). Samples results are presented in Figure 4.11. As the bit rate increases, the Sparseland model becomes less relevant, as the representations become non-sparse and this results with a deterioration in the performance.

<table>
<thead>
<tr>
<th>Dictionary</th>
<th>Bits per Coefficients</th>
<th>PSNR</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-SVD dictionary</td>
<td>8</td>
<td>34.1564</td>
<td>0.70651</td>
</tr>
<tr>
<td>Overcomplete DCT</td>
<td>8</td>
<td>32.4021</td>
<td>0.69419</td>
</tr>
<tr>
<td>Complete DCT dictionary</td>
<td>8</td>
<td>32.3917</td>
<td>0.70302</td>
</tr>
</tbody>
</table>

Figure 4.11: Sample compression results.
Chapter 5

K-SVD-Based Image Denoising

5.1 Introduction

In this section we address the classic image denoising problem: An ideal image \( z \) is measured in the presence of an additive zero-mean white and homogeneous Gaussian noise, \( v \), with standard deviation \( \sigma \). The measured image, \( y \), is thus

\[
y = z + v.
\]

We desire to design an algorithm that can remove the noise from \( y \), getting as close as possible to the original image, \( z \).

The image denoising problem is important, not only because of the evident applications it serves. Being the simplest possible inverse problem, it provides a convenient platform over which image processing ideas and techniques can be assessed. Indeed, numerous contributions in the past 50 years or so addressed this problem from many and diverse points of view. Statistical estimators of all sorts, spatial adaptive filters, stochastic analysis, partial differential equations, transform-domain methods, splines and other approximation theory methods,
morphological analysis, order statistics, and more, are some of the many directions explored in studying this problem. In this work we have no intention to provide a survey of this vast activity. Instead, we intend to concentrate on applying the main concepts this thesis deals with for image denoising: the use of sparse and redundant representations over trained dictionaries. We will use the K-SVD within this algorithm, showing that it leads to state-of-the-art results. This way we intend to strengthen the general claim made in this thesis, which states “The Sparseland model for images, and the K-SVD as the means to get a sparsifying dictionary, are both promising methods that can lead to better performance in many applications in signal and image processing”.

Using redundant representations and sparsity as driving forces for denoising of signals has drawn a lot of research attention in the past decade or so. At first, sparsity of the unitary wavelet coefficients was considered, leading to the celebrated shrinkage algorithm [31, 22, 34, 32, 33, 95, 9, 78, 62]. One reason to turn to redundant representations was the desire to have the shift invariance property [12]. Also, with the growing realization that regular separable 1D wavelets are inappropriate for handling images, several new tailored multiscale and directional redundant transforms were introduced, including the curvelet [7, 8], contourlet [18, 19], wedgelet [23], bandlet [74, 73], and the steerable wavelet [46, 97]. In parallel, the introduction of the matching pursuit [75, 84] and the basis pursuit denoising [11], gave rise to the ability to address the image denoising problem as a direct sparse decomposition technique over redundant dictionaries. All these lead to what is considered today as some of the best available image denoising methods – see [87, 99, 43, 77] for few representative works.

In addressing general inverse problems in image processing using the Bayesian
approach, an image prior is necessary. Traditionally, this has been handled by choosing a prior based on some simplifying assumptions, such as spatial smoothness, min/max-entropy, or sparsity in some transform domain. While these common approaches lean on a guess of a mathematical expression for the image prior, we concentrate here on example-based restoration techniques, which suggest to learn the prior from images somehow. For example, assuming a spatial smoothness-based Markov random field prior of a specific structure, one can still question (and thus train) the derivative filters to apply on the image, and the robust function to use in weighting these filters’ outcome [108, 58, 91].

When this prior-learning idea is merged with sparsity and redundancy, we return to the main motive of this thesis - training the dictionary to be used for the restoration. In this work we consider two options: (i) training the dictionary using patches from the corrupted image itself; or (ii) training on a corpus of patches taken from a high-quality set of images.

In this work we use the K-SVD algorithm described above because of its simplicity and efficiency for this task. Also, due to its structure, we shall see how the training and the denoising fuse together naturally into one coherent and iterated process, when training is done on the given image directly.

Since dictionary learning algorithms, and as such, the K-SVD, are limited in handling small image patches, a natural difficulty arises: How can we use it for general images of arbitrary size? In this work we propose a global image prior that forces sparsity over patches in every location in the image (with overlaps). This aligns with a similar idea, appearing in [91], for turning a local MRF-based prior into a global one. We define a maximum a-posteriori probability (MAP) estimator as the minimizer of a well-defined global penalty term. Its numerical solution leads
to a simple iterated patch-by-patch sparse coding and averaging algorithm, that is closely related to the ideas explored in [55, 56, 57], and generalizes them.

When considering the available global and multiscale alternative denoising schemes (e.g., based on curvelet, contourlet, and steerable wavelet), it looks like there is much to be lost in working on small patches. Is there any chance of getting a comparable denoising performance with a local-sparsity based method? In that respect, the image denoising work reported in [87] is of great importance. Beyond the specific novel and highly effective algorithm described in that paper, Portilla and his co-authors posed a clear set of comparative experiments that standardize how image denoising algorithms should be assessed and compared one versus the other. We make use of these exact experiments and show that the newly proposed algorithm performs similarly, and often better, compared to the denoising performance reported in their work. This puts this proposed algorithm in the front row of image processing techniques.

To summarize, the novelty of this work includes the way we use local sparsity and redundancy as ingredients in a global Bayesian objective – this part is described in the next Section, along with its emerging iterated numerical solver. Also novel in this work is the idea to train dictionaries for the denoising task, rather than use pre-chosen ones. As already mentioned earlier, when training is done on the corrupted image directly, the overall training-denoising algorithm becomes fused into one iterative procedure that comprises of steps of denoising of the image, followed by an update of the dictionary, as the K-SVD does it. This is described in Section 5.3 in details. In Section 5.4 we show some experimental results that demonstrate the effectiveness of this algorithm.
5.2 From Local to Global Bayesian Reconstruction

In this section we start the presentation of the proposed denoising algorithm by first introducing how sparsity and redundancy are brought to use. For this, we return to the Sparseland model described in Section 1.3 and adopt it for image patches. Once this is set, we will discuss how local treatment on those image patches turns into a global prior in a Bayesian reconstruction framework.

5.2.1 The Sparseland Model for Image Patches

In this section we return to the Sparseland model described in 1.3, and apply it on image patches. We consider image patches of size $\sqrt{n} \times \sqrt{n}$ pixels, ordered as column vectors $z \in \mathbb{R}^n$. For the construction of the Sparseland model, we need to define a dictionary (matrix) of size $D \in \mathbb{R}^{n \times K}$ (with $K > n$, implying that it is redundant). At the moment we shall assume that this matrix is known and fixed. Put loosely, the proposed model suggests that every image patch, $z$, could be represented sparsely over this dictionary, i.e., the solution of

$$\hat{x} = \arg \min_x \|x\|_0 \text{ subject to } Dx \approx z, \quad (5.2)$$

is indeed very sparse, $\|\hat{x}\|_0 \ll n$.

This model should be made more precise by replacing the rough constraint $Dx \approx z$ with a clear requirement to allow a bounded representation error, $\|Dx - z\|_2 \leq \epsilon$. Also, one needs to define how deep is the required sparsity, adding a requirement of the form $\|\hat{x}\|_0 \leq L \ll n$, that states that the sparse representation uses no more than $L$ atoms from the dictionary for every image patch instance. With the triplet $(\epsilon, L, D)$ in place, our model is well-defined.

Now assume that $z$ indeed belongs to the $(\epsilon, L, D)$-Sparseland signals. Con-
sider a noisy version of it, \( y \), contaminated by an additive zero-mean white Gaussian noise with standard deviation \( \sigma \). The MAP estimator for denoising this image patch is built by solving

\[
\hat{x} = \arg \min_x \|x\|_0 \quad \text{subject to} \quad \|Dx - y\|_2^2 \leq T,
\]

where \( T \) is dictated by \( \epsilon \) and \( \sigma \). The denoised image is thus given by \( \hat{z} = D\hat{x} \) [11, 28, 105]. Notice that the above optimization task can be changed to be

\[
\hat{x} = \arg \min_x \|Dx - y\|_2^2 + \mu \|x\|_0,
\]

so that the constraint becomes a penalty. For a proper choice of \( \mu \) the two problems are equivalent. We will use this alternative terminology from now on, as it makes the presentation of later parts simpler to follow. Naturally, the above problem is solved approximately using of the pursuit techniques mentioned earlier. Here, as in the previous chapter, we adopt the OMP due to its simplicity.

### 5.2.2 From Local Analysis to a Global Prior

If we want to handle a larger image, \( Z \), of size \( \sqrt{N} \times \sqrt{N} \) \( (N \gg n) \), and we are still interested in using the above described model, one option is to redefine the model with a larger dictionary. Indeed, when using this model with a dictionary emerging from the contourlet or curvelet transforms, such scaling is simple and natural [77].

However, when we insist on using a specific fixed and small size dictionary \( D \in \mathcal{R}^{n \times K} \), this option no longer exists. Thus, a natural question arises concerning the use of such a small dictionary in the first place. Two reasons come to mind:

(i) When training takes place (as we will show in the next section), only small
dictionaries can be composed; and furthermore, (ii) A small dictionary implies a locality of the resulting algorithms, which simplifies the overall image treatment.

We next describe possible ways to use such a small dictionary when treating a large image. A heuristic approach is to work on smaller patches of size $\sqrt{n} \times \sqrt{n}$ and tile the results. In doing so, visible artifacts may occur on block boundaries. One could also propose to work on overlapping patches and average the results in order to prevent such blockiness artifacts, as indeed practiced in [55, 56, 57]. As we shall see next, a systematic global approach towards this problem leads to this very option as a core ingredient in an overall algorithm.

If our knowledge on the unknown large image $Z$ is fully expressed in the fact that every patch in it belongs to the $(\epsilon, L, D)$-Sparseland model, then the natural generalization of the above MAP estimator is the replacement of (5.4) with

$$\{x_{ij}, \hat{Z}\} = \arg\min_{x_{ij}, Z} \lambda||Z - Y||_2^2 + \sum_{ij} \mu_{ij}||x_{ij}||_0 + \sum_{ij} ||Dx_{ij} - R_{ij}Z||_2^2.$$ (5.5)

In this expression the first term is the log-likelihood global force that demands the proximity between the measured image, $Y$, and its denoised (and unknown) version $Z$. Put as a constraint, this penalty would have read $||Z - Y||_2^2 \leq \text{Const} \cdot \sigma^2$, and this reflects the direct relationship between $\lambda$ and $\sigma$. The second and the third terms are the image prior that makes sure that in the constructed image, $Z$, every patch $z_{ij} = R_{ij}Z$ of size $\sqrt{n} \times \sqrt{n}$ in every location (thus the summation by $i, j$) has a sparse representation with bounded error. Similar conversion has also been practiced by Roth and Black when handling an MRF prior [91].

The matrix $R_{ij}$ is an $n \times N$ matrix that extracts the $(ij)$ block from the image. For an $\sqrt{N} \times \sqrt{N}$ image $Z$, the summation over $i, j$ includes $(\sqrt{N} - \sqrt{n} + 1)^2$ items, considering all image patches of size $\sqrt{n} \times \sqrt{n}$ in $Z$ with overlaps. As to the coefficients $\mu_{ij}$, those must be location dependent, so as to comply with a set
of constraints of the form \(\|Dx_{ij} - z_{ij}\|^2 \leq T\).

### 5.2.3 Numerical Solution

When the underlying dictionary \(D\) is assumed known, the proposed penalty term in (5.5) has two kinds of unknowns: the sparse representations \(\hat{x}_{ij}\) per each location, and the overall output image \(Z\). Instead of addressing both together, we propose a block-coordinate minimization algorithm that starts with an initialization \(Z = Y\), and then seeks the optimal \(\hat{x}_{ij}\). In doing so, we get a complete decoupling of the minimization task to many smaller ones, each of the form

\[
\hat{x}_{ij} = \arg\min_{x} \mu_{ij} \|x\|_0 + \|Dx - z_{ij}\|^2_2,
\]

(5.6)

handling one image patch. Solving this using the OMP is easy, gathering one atom at a time, and stopping when the error \(\|Dx - z_{ij}\|^2_2\) goes below \(T\). This way, the choice of \(\mu_{ij}\) has been handled implicitly. Thus, this stage works as a sliding window sparse coding stage, operated on each block of \(\sqrt{n} \times \sqrt{n}\) at a time.

Given all \(\hat{x}_{ij}\), we can now fix those and turn to update \(Z\). Returning to (5.5), we need to solve

\[
\hat{Z} = \arg\min_{Z} \lambda \|Z - Y\|^2_2 + \sum_{ij} \|D\hat{x}_{ij} - R_{ij}Z\|^2_2.
\]

(5.7)

This is a simple quadratic term that has a closed-form solution of the form

\[
\hat{Z} = \left(\lambda I + \sum_{ij} R_{ij}^T R_{ij}\right)^{-1} \left(\lambda Y + \sum_{ij} R_{ij}^T D\hat{x}_{ij}\right).
\]

(5.8)

This rather cumbersome expression may mislead, as all it says is that averaging of the denoised patches is to be done, with some relaxation obtained by averaging with the original noisy image. The matrix to invert in the above expression is a
diagonal one, and thus the calculation of (5.8) can be also done on a pixel-by-pixel basis, following the previously described sliding window sparse coding steps.

So far we have seen that the denoising algorithm calls for sparse coding of small patches, and an averaging of their outcomes. However, if minimization of (5.5) is our goal, then this process should proceed. Given the updated $Z$, we can repeat the sparse coding stage, this time working on patches from the already denoised image. Once this is done, a new averaging should be calculated, and so on, and so forth. Thus, we obtain exactly what Guleryuz suggested in his work – iterated denoising via sparse representation, and we may regard the analysis proposed here as a rigorous way to justify such an iterated scheme [55, 56, 57].

5.3 Example-Based Sparsity and Redundancy

The entire discussion so far has been based on the assumption that the dictionary $D \in \mathcal{R}^{n \times K}$ is known. We can certainly make some educated guesses as to which dictionaries to use. In fact, following Guleryuz's work, the DCT seems like a plausible choice [55, 56, 57]. Indeed, we might do better by using a redundant version of the DCT, as practiced in Section 4.3. Still, the question remains: can we make a better choice for $D$ based on training? In light of the previous chapter discussion, the answer is clearly positive, as we describe shortly. We start with the simpler (and less effective) option of training the dictionary with the K-SVD on a set of image patches taken from good quality images, and then turn to discuss the option of training on the corrupted image itself.
5.3.1 Training on the Corpus of Image Patches

Given a set of image patches $\mathcal{Z} = \{z_j\}_{j=1}^{M}$, each of size $\sqrt{n} \times \sqrt{n}$, and assuming that they emerge from a specific $(\epsilon, L, D)$-Sparseland model, we would like to estimate this model parameters, $(\epsilon, L, D)$. Put formally, we seek the dictionary $D$ that minimizes

$$
\varepsilon(D, \{x_j\}_{j=1}^{M}) = \sum_{j=1}^{M} [\mu_j \|x_j\|_0 + \|Dx_j - z_j\|_2^2].
$$

(5.9)

Just as before, the above expression seeks to get a sparse representation per each of the examples in $\mathcal{Z}$, and obtain a small representation error. The choice for $\mu_j$ dictates how those two forces should be weighted, so as to make one of them a clear constraint. For example, constraining $\forall j \|x_j\|_0 = L$ implies specific values for $\mu_j$, while requiring $\forall j \|Dx_j - z_j\|_2^2 \leq \epsilon^2$ leads to others.

The K-SVD algorithm described in Section 4 proposes an iterative algorithm designed to handle the above task effectively [1, 2]. Adopting again the block-coordinate descent idea, the computations of $D$ and $\{x_j\}_{j=1}^{M}$ are separated. Assuming that $D$ is known, the penalty posed in Equation (5.9) reduces to a set of $M$ sparse coding operations, very much like the ones seen in Equation (5.6). Thus, OMP can be used again to obtain the near-optimal (recall that OMP is an approximation algorithm, and thus a true minimization is not guaranteed) set of representation vectors $\{x_j\}_{j=1}^{M}$.

Assuming these representation vectors fixed, the K-SVD proposes an update of the dictionary one column at a time. Each such update can be done optimally by performing a singular value decomposition (SVD) operation on residual data matrices, computed only on the examples that use this atom. This way, the value of $\varepsilon(D, \{x_j\}_{j=1}^{M})$ is guaranteed to drop per an update of each dictionary atom,
and along with this update, the representation coefficients change as well (see Section 4 for more details).

When adopted to the denoising task at hand, a crucial step is the choice of the examples to train on. Is there really a universal dictionary that fits all images well? If there is one, which examples shall we use to find it? The experiments that follow in the next section bring us to the conclusion that while a reasonably good dictionary that fits all is indeed within reach, extracting state-of-the-art denoising performance calls for a more complex model that uses several dictionaries switched by content – an option we do not explore in this work.

Also, since the penalty minimized here in Equation (5.9) is a highly non-convex functional, sub-optimal solutions are likely to haunt us. Thus, a wise initialization could be of great worth. In our experiments we started with the already mentioned redundant DCT, which proves to be a good dictionary choice. This also enabled us to apply fewer iterations.

Another puzzling issue is the redundancy factor $K/n$ – how should we choose $K$, the number of columns in $D$? Is there an optimal choice? In this work we do not address this important question, and simply choose a value we find empirically to perform well. Further work is required to explore this matter.

\subsection*{5.3.2 Training on the Corrupted Image}

Instead of supplying an artificial set of examples to train on, as proposed above, one could take the patches from the corrupted image, $Z = \{y_j\}_{j=1}^M$, where $M = (\sqrt{N} - \sqrt{n} + 1)^2$. Since the K-SVD dictionary learning process has in it a noise rejection capability (see experiments reported in Section 4.3), this seems like a natural idea. Furthermore, rather than using unrelated examples that call
for the universality assumption of the *Sparseland* model, this option tailors the
dictionary to the image treated.

At first sight, this change in the origin of the examples to train on seems
to be of technical worth, and has no impact on the overall algorithm. However, a
close inspection of both the functional \( \epsilon(\{D, \{x_j\}_{j=1}^M\} \) in (5.9), and the global MAP
penalty in (5.5), reveals the close resemblance between the two. This implies that
the dictionary design could be embedded within the Bayesian approach. Returning
to Equation (5.5), we can regard also \( D \) as an unknown, and define our problem
as

\[
\{ \hat{D}, \hat{x}_{ij}, \hat{Z} \} = \arg \min_{D, x_{ij}, Z} \lambda \|Z - Y\|_2^2 + \sum_{ij} \mu_{ij} \|X_{ij}\|_0 + \sum_{ij} \|Dx_{ij} - R_{ij}Z\|_2^2. \tag{5.10}
\]

Following the previously constructed algorithm, we can assume a fixed \( D \) and \( Z \),
and compute the representations \( \hat{x}_{ij} \). This requires, as before, a sparse coding
stage that deploys the OMP. Given those representations, the dictionary can be
now updated, using a sequence of \( K \) SVD operations.

Once done, the output image can be computed using (5.8). However, an
update of the output image \( Z \) changes the noise level \( \sigma \), which up until now has
been considered as known, and was used in the preceding two stages. Therefore,
we choose to perform several more iterations of representation computation and
dictionary update, using the same value of \( \sigma \), before finding the output image \( Z \).
This algorithm is described in detail in Figure 5.1.

In evaluating the computational complexity of this algorithm, we consider
all three stages - sparse coding (OMP process), dictionary update (these stages are
iterated \( J \) times), and final averaging process. All stages can be done efficiently,
requiring \( O(nKLM) \) operations per pixel, where \( n \) is the block dimension, \( K \) is
the number of atoms in the dictionary, and \( L \) is the number of nonzero elements in each coefficient vector. \( L \) depends strongly on the noise level, e.g., for \( \sigma = 10 \), the average \( L \) is 2.96, and for \( \sigma = 20 \), the average \( L \) is 1.12.

5.4 Results

In this section we demonstrate the results achieved by applying the above methods on several test images, and with several dictionaries. The tested images, as also the tested noise levels, are all the same as those used in the denoising experiments reported in [87], in order to enable a fair comparison.

Table 5.1 summarizes these denoising results for the DCT dictionary, the globally trained dictionary, and training on the corrupted images directly (referred to hereafter as the adaptive dictionary). In most of these experiments, the dictionaries used were of size 64 \( \times \) 256, designed to handle image patches of size 8 \( \times \) 8 pixels \( (n = 64, K = 256) \). Only in the stronger noise levels \( (\sigma = 50, 75, 100) \), while applying the adaptive dictionary method, did we use larger patches of size 16 \( \times \) 16, resulting a dictionary of size 256 \( \times \) 256. Every result reported is an average over 5 experiments with different realizations of the noise.

The redundant DCT dictionary is described in Figure 5.2, with each of its atoms shown as an 8 \( \times \) 8 pixel image. This dictionary was also used as the initialization for all the training algorithms that follow. The globally trained dictionary is shown in Figure 5.3. This dictionary was produced by the K-SVD algorithm (executed 180 iterations, using OMP for sparse coding with \( L = 6 \)), trained on a data-set of 100,000 8 \( \times \) 8 patches (very much like the one used in Section 4.3). Those patches were taken from an arbitrary set of clean natural images (unrelated to the test images), some of which are shown in Figure 5.4.
Task: Denoise a given image \( Y \) from white and additive Gaussian white noise with standard deviation \( \sigma \).

Algorithm Parameters: \( n \) - block size, \( K \) - dictionary size, \( J \) - number of training iterations, \( \lambda \) - Lagrange multiplier, and \( C \) - noise gain.

\[
\min_{Z,D,A} \left\{ \lambda \| Y - Z \| + \sum_{ij} \mu_{ij} \| x_{ij} \|_0 + \sum_{ij} \| D x_{ij} - R_{ij} Z \|_2 \right\}
\]

(1) Initialization: Set \( Z = Y \), \( D = \) overcomplete DCT dictionary.

(2) Repeat \( J \) times:

- **Sparse Coding Stage:** Use any pursuit algorithm to compute the representation vectors \( x_{ij} \) for each patch \( R_{ij} Z \), by approximating the solution of

  \[
  \forall_{ij} \min_{x_{ij}} \| x_{ij} \|_0 \quad \text{s.t.} \quad \|R_{ij} Z - Dx_{ij}\|_2^2 \leq (C\sigma)^2.
  \]

- **Dictionary Update Stage:** For each column \( l = 1,2,\ldots,K \) in \( D \), update it by

  * Find the set of patches that use this atom, \( \omega_l = \{(i,j)|x_{ij}(l) \neq 0\} \).
  * For each index \((i,j) \in \omega_l\), compute its representation error

    \[
e_{ij}^l = R_{ij} Z_{ij} - \sum_{m \neq l} d_m x_{ij}(m).
    \]

  * set \( E_l \) as the matrix whose columns are \( \{e_{ij}^l\}_{(i,j) \in \omega_l} \)

  * Apply SVD decomposition \( E_l = U \Delta V^T \). Choose the updated dictionary column \( d_l \) to be the first column of \( U \). Update the coefficient values \( \{x_{ij}(l)\}_{(i,j) \in \omega_l} \) to be the entries of \( V \) multiplied by \( \Delta(1,1) \).

(3) Set:

\[
Z = \left( \lambda I + \sum_{ij} R_{ij}^T R_{ij} \right)^{-1} \left( \lambda Y + \sum_{ij} R_{ij}^T D x_{ij} \right)
\]

Figure 5.1: Denoising Procedure using a dictionary trained on patches from the corrupted image. For our experiments with noise level lower than \( \sigma = 50 \), we used the OMP pursuit method, and set \( J = 10, \lambda = 30/\sigma \) and \( C = 1.15 \).
In all experiments, the denoising process included a sparse-coding of each patch of size $8 \times 8$ pixels from the noisy image. Using the OMP, atoms were accumulated till the average error passed the threshold, chosen empirically to be $\epsilon = 1.15 \cdot \sigma$. This means that our algorithm assumes the knowledge of $\sigma$ — very much like assumed in [87]. The denoised patches were averaged, as described in Equation (5.8), using $\lambda = 30/\sigma$ (see below for an explanation for this choice of $\lambda$).

As was said before, we chose to apply only one iteration in the iterative process suggested previously in 5.2.3. Following iterations require knowledge of the new noisy parameter $\sigma$, which is unknown after first changing $Z$.

When training the dictionary on overlapping patches from the noisy image itself, each such experiment included $(256 - 7)^2 = 62,001$ patches (all available patches from the 256 $\times$ 256 images, and every second patch from every second row in the 512 $\times$ 512 size images). The algorithm described in details in Figure 5.1 was applied. In this denoising method, the parameters for the stronger noise levels ($\sigma = 50, 75, 100$) were a bit modified. The patches were set to size $16 \times 16$, resulting a dictionary of size $256 \times 256$. The threshold parameter was set to $\epsilon = 1.05 \cdot \sigma$, and $\lambda = 0$.

As can be seen from Table 5.1, the results of all methods are very close to each other in general. Averaging the results that correspond to [87] in this table for noise levels lower than $\sigma = 50$, the value is 34.62dB. A similar averaging over the DCT dictionary results gives 34.45dB, implying an average difference of 0.17dB, in favor of Portilla’s method. This is the same case with the globally trained dictionary, which means that our attempt to train one global dictionary for

---

1 The strong noise experiments are problematic to analyze, because clipping of the dynamic range to $[0, 255]$, as often done, causes a severe deviation from the Gaussian distribution model assumed.
<table>
<thead>
<tr>
<th>$\sigma/\text{PSNR}$</th>
<th>Lena</th>
<th>Barb</th>
<th>Boats</th>
<th>Fgrpt</th>
<th>House</th>
<th>Peppers</th>
<th>Average</th>
<th>$\sigma_{\text{PSNR}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/42.11</td>
<td>43.23</td>
<td>43.55</td>
<td>43.29</td>
<td>43.61</td>
<td>43.05</td>
<td>42.92</td>
<td>44.07</td>
<td>43.30</td>
</tr>
<tr>
<td></td>
<td>43.23</td>
<td>43.58</td>
<td>43.10</td>
<td>43.67</td>
<td>41.86</td>
<td>43.14</td>
<td>42.94</td>
<td>42.99</td>
</tr>
<tr>
<td>5/34.15</td>
<td>38.49</td>
<td>38.51</td>
<td>37.79</td>
<td>37.93</td>
<td>36.68</td>
<td>36.48</td>
<td>38.65</td>
<td>39.07</td>
</tr>
<tr>
<td></td>
<td>38.48</td>
<td>38.60</td>
<td>37.32</td>
<td>38.08</td>
<td>36.64</td>
<td>37.22</td>
<td>38.86</td>
<td>39.37</td>
</tr>
<tr>
<td>10/28.13</td>
<td>35.61</td>
<td>35.28</td>
<td>34.03</td>
<td>33.97</td>
<td>32.45</td>
<td>32.14</td>
<td>35.35</td>
<td>35.41</td>
</tr>
<tr>
<td></td>
<td>35.40</td>
<td>35.47</td>
<td>33.07</td>
<td>34.42</td>
<td>33.53</td>
<td>33.64</td>
<td>35.69</td>
<td>35.98</td>
</tr>
<tr>
<td>15/24.61</td>
<td>33.90</td>
<td>33.38</td>
<td>31.86</td>
<td>31.63</td>
<td>30.14</td>
<td>29.71</td>
<td>33.64</td>
<td>33.49</td>
</tr>
<tr>
<td></td>
<td>33.60</td>
<td>33.70</td>
<td>30.61</td>
<td>32.37</td>
<td>31.70</td>
<td>31.38</td>
<td>34.03</td>
<td>34.32</td>
</tr>
<tr>
<td></td>
<td>32.66</td>
<td>32.00</td>
<td>30.32</td>
<td>30.83</td>
<td>30.38</td>
<td>29.91</td>
<td>32.81</td>
<td>33.20</td>
</tr>
<tr>
<td>20/22.11</td>
<td>32.27</td>
<td>32.38</td>
<td>28.87</td>
<td>30.38</td>
<td>28.60</td>
<td>28.01</td>
<td>30.31</td>
<td>30.20</td>
</tr>
<tr>
<td></td>
<td>31.69</td>
<td>30.89</td>
<td>29.13</td>
<td>28.65</td>
<td>29.37</td>
<td>28.78</td>
<td>31.40</td>
<td>31.03</td>
</tr>
<tr>
<td></td>
<td>31.20</td>
<td>31.32</td>
<td>27.57</td>
<td>29.60</td>
<td>29.17</td>
<td>29.28</td>
<td>31.82</td>
<td>32.15</td>
</tr>
<tr>
<td>25/20.17</td>
<td>28.61</td>
<td>27.44</td>
<td>25.48</td>
<td>24.75</td>
<td>26.38</td>
<td>25.57</td>
<td>28.16</td>
<td>22.01</td>
</tr>
<tr>
<td></td>
<td>27.77</td>
<td>28.21</td>
<td>24.06</td>
<td>26.28</td>
<td>25.91</td>
<td>26.05</td>
<td>22.68</td>
<td>23.90</td>
</tr>
<tr>
<td>50/14.15</td>
<td>26.84</td>
<td>25.63</td>
<td>23.65</td>
<td>22.83</td>
<td>24.79</td>
<td>23.85</td>
<td>26.41</td>
<td>25.10</td>
</tr>
<tr>
<td></td>
<td>25.81</td>
<td>26.27</td>
<td>22.54</td>
<td>24.20</td>
<td>24.02</td>
<td>24.35</td>
<td>19.73</td>
<td>21.99</td>
</tr>
<tr>
<td>75/10.63</td>
<td>25.64</td>
<td>24.42</td>
<td>22.61</td>
<td>21.89</td>
<td>23.75</td>
<td>22.79</td>
<td>21.22</td>
<td>17.99</td>
</tr>
<tr>
<td></td>
<td>24.45</td>
<td>24.85</td>
<td>21.73</td>
<td>22.63</td>
<td>22.83</td>
<td>23.13</td>
<td>18.23</td>
<td>20.36</td>
</tr>
</tbody>
</table>

Table 5.1: Summary of the denoising PSNR results in [dB]. In each cell four denoising results are reported. Top left: results of Portilla et al. [87], Top right: overcomplete DCT, Bottom left: global trained dictionary, Bottom right: adaptive dictionary trained on noisy image. In each such set we highlighted the best result. All numbers are an average over 5 experiments. The last two columns present the average results over all images and their variance.
images performs as good as the fixed redundant DCT. However, for the method of the image-adaptive dictionary, an average of 34.86dB is obtained, giving an average advantage of 0.24dB over Portilla’s method.

![Figure 5.2: The overcomplete DCT dictionary.](image)

In order to better visualize the results and their comparison to those in [87], Figure 5.5 presents the difference of the denoising results of the two proposed methods and the overcomplete DCT compared with those of [87] (which appears as a zero straight reference line). This comparison is presented for the images ‘Peppers’, ‘House’ and ‘Barbara’. Notice that for these images, the adaptive dictionary outperforms the reported results of Portilla et al. for all noise levels lower than \( \sigma = 75 \), while the global dictionary often achieves very close results. In the image ‘Barbara’, however, which contains high-frequency texture areas, the adaptive dictionary that learns the specific characteristics has a clear advantage over the globally trained dictionary.

Figure 5.6 further describes the behavior of the denoising algorithm that uses
Figure 5.3: The globally trained dictionary.

Figure 5.4: Sample from the images used for training the global dictionary.
Figure 5.5: Comparison between the three presented methods (Overcomplete DCT, global trained dictionary and adaptive dictionary trained on patches from the noisy image) and the results achieved recently in [87] for three test images.
the adaptive dictionary. Each K-SVD iteration improves the denoising results, with the initial dictionary set to be the overcomplete DCT. A graph presenting this consistent improvement for several noise levels is presented in this Figure. All graphs show the improvement over the first iteration, and therefore all curves start at zero, going towards positive values. As can be seen, a gain of up to 1dB is achievable. Figure 5.7 shows the results of the proposed algorithms for the image ‘Barbara’, and for $\sigma = 20$. The final adaptive dictionary that leads to those results is presented in Figure 5.8.

![Graph showing improvement in denoising results](image)

**Figure 5.6:** The improvement in the denoising results after each iteration of the K-SVD algorithm, executed on noisy patches of the image ‘Peppers’.

We now turn to study the effect of the parameter $\lambda$ in Equation (5.8). As expected, we found that a proper choice for $\lambda$ is dependent on the noise level. As the noise increases, better results are achieved with small values of $\lambda$ and vice-versa. This is indeed expected, as relatively ‘clean’ images should have a stronger
Figure 5.7: Example of the denoising results for the image 'Barbara' with $\sigma = 20$ – the original, the noisy, and two restoration results.
Figure 5.8: Example of the denoising results for the image ‘Barbara’ with $\sigma = 20$ – the adaptively trained dictionary.
effect on the outcome, while very noisy ones should effect the outcome weakly, if at all. We tested several values for this parameter, and found empirically that the best results are achieved with $\lambda \approx 30/\sigma$. It is interesting to see that all three denoising methods (overcomplete DCT, global dictionary, and adaptive dictionary trained on noisy patches), and all noise levels generally agree with this choice. In Figure 5.9 we present the improvement (and later, deterioration) achieved when increasing the value of $\lambda$ in the averaging process (Equation (5.8)). In this Figure one image (‘Peppers’) was tested with four noise levels ($\sigma = 5, 10, 20, 50$) and with all three methods, resulting with 12 curves. The choice $\lambda = 30/\sigma$ seems to be near the peak for all these graphs.

To conclude this experimental section, we refer to our arbitrary choice of $K = 256$ dictionary atoms (this choice had an effect over all three experimented methods). We conducted another experiment, which compares between several values of $K$. In this experiment, we tested the denoising results of the three proposed methods on the image ‘House’ for an initial noise level of $\sigma = 15$ (24.61dB) and $\lambda = 30/\sigma$. The tested redundancy values (of $K$) were 64, 128, 256, and 512. The average results of 4 executions (per each test) are presented in Figure 5.10. As can be seen, the increase of the number of dictionary elements generally improves the results, although this improvement is small ($0 - 0.16$dB). This increase is most effective in the adaptive dictionary method.

5.5 Summary

This chapter presented a relatively simple method for image denoising, leading to state-of-the-art performance, equivalent and sometimes surpassing recently published leading alternatives. The proposed method is based on local opera-
Figure 5.9: The improvement (and later, deterioration) of the denoising results when increasing the value of $\lambda$ in the averaging process in Equation (5.8).

Figure 5.10: The effect of changing the number of dictionary elements ($K$) on the final denoising results for the image 'House' and for $\sigma = 15$. 
tions and involves sparse decompositions of each image block under one fixed over-complete dictionary, and a simple average calculations. The content of the dictionary is of prime importance for the denoising process – we have shown that a dictionary trained for natural real images, as well as an adaptive dictionary trained on patches of the noisy image itself, both perform very well. The use of the K-SVD, along with the reliance on sparsity and redundancy as guiding models, have both led us to obtain excellent results.
Chapter 6

Constrained K-SVD

6.1 General Motivation

Most of the proposed methods for dictionary learning are designed to produce general dictionaries without any constraints other than the number of atoms. However, additional constraints could emerge from a prior knowledge on the nature of the data, which enforces properties on the underlying dictionary. Furthermore, structural constraints on the dictionary can lead to a more efficient usage, and in general, may be required for several applications due to computational considerations.

Considering all the dictionary training algorithms mentioned in section 2.3, the K-SVD algorithm is different from the others because it updates one atom at a time, and in each such update allows changing the relevant coefficients. Each update is a convex problem with a simple solution. This fact makes the K-SVD attractive for the development of different variations, which introduce additional constraints.

In this section we refer to the problem of learning dictionaries under structural constraints by suggesting variations of the K-SVD algorithm. The first constraint we refer to is the non-negativity. We allow the dictionary to consist of only
non-negative elements. Often (although not necessary) we restrict non-negativity also on the coefficients, and solve a variation of problem (4.4),

$$\min_D \sum_i \|y_i - Dx_i\|^2 \quad s.t. \quad \forall i \|x_i\|_0 \leq T_0, \ x_i \geq 0, \ d_i \geq 0,$$

where in $v \geq 0$ we mean that all entries of the vector $v$ are greater or equal zero.

The second variation we consider is the creation of a shift-invariant dictionary. We train a dictionary $D \in \mathbb{R}^{m \times L}$, for $m \leq n$. Each signal $y$ is represented as a sum of shifted atoms $S_t d_i$, for $S_t$ a matrix that locates a vector of size $m$ in the $t$'th location of a vector of size $n$. As a result, the number of representing atoms is $L \cdot s$, where $s$ is the number of possible shifts of each atom.

In fact, the above could be generalized by considering a more general constraint that forces each atom in the dictionary to be the product of some known matrix $R \in \mathbb{R}^{n \times m}$ with other unconstrained vector $\tilde{d}$, such that $d = R\tilde{d}$. When $\text{rank}(R) < n$ it becomes a constraint, and the real variables to discover are the entries of $\tilde{d}$. We present a way to extend the K-SVD to address such a structure.

Up until now, representations of high dimensional signals by trained dictionaries were done by representing small patches of those signals, as practiced in the previous chapter [1, 82]. Computational considerations, as well as memory constraints, prevent the K-SVD (and other methods) from learning dictionaries for such signals. In an attempt to overcome this difficulty, we consider the incorporation of linear constraints within the K-SVD, in a way which can lead to a multiscale version of it, and this way handle signals of arbitrary size.

Finally, we present a totally different approach for structuring the dictionary. Instead of being a set of distinct atoms, we propose an 'Image signature Dictionary' (ISD), where the dictionary is a matrix in which each patch can serve as a representing atom. We show that such a scheme obtains shift-invariant prop-
erties, and overcomes other difficulties that arise from decomposition under a shift-invariant dictionary. Moreover, such an approach requires much less memory than a ‘regular’ dictionary, and potentially much less computational effort for the decomposition.

For clarity and simplicity, we adopt in this chapter two definitions.

- We use the set symbol \( \{ v_i \}_{i=1}^{N} \) to represent a matrix that contains \( v_1, v_2, \ldots, v_N \) as its columns.

- For a matrix \( A \) and a set of indices \( \Omega \), we define \( A^\Omega \) as the matrix \( A \), restricted only to the columns whose indices are in \( \Omega \).

### 6.2 Non-Negative K-SVD (NN-K-SVD)

#### 6.2.1 General

For some applications, using sparse representations and overcomplete dictionaries together with forcing non-negativity on both the dictionary and the coefficients, may lead to a better modeling of the problem, and as such to an effective method for revealing the ‘ingredients’ from which all training signals are built of [60, 61]. The inability to subtract values from the linear combination forces the dictionary elements to become sparser, and converge to the building blocks of the training signals. This subject is often referred to in literature as Non-Negative Matrix Factorization, or NMF, computing both the dictionary and the coefficient matrices, whose product approximates the signal matrix \( Y \approx DX \). Application for NMF are many and include dimensionality reduction [5] and analysis of data such as audio [98], text [94], and data obtained from astronomical spectrometers [86].
In what follows we describe sparse non-negative decomposition under overcomplete dictionaries. Then, we review previously reported methods for learning non-negative dictionaries and describe the K-SVD variation for this task (referred to as NN-K-SVD). Finally, we conduct some validation tests and prove the advantage of NN-K-SVD.

### 6.2.2 Non-negative Decomposition

Pursuit methods with non-negativity constraints are similar to those presented earlier. A non-negative version of BP minimizes the following convex function [60],

$$
\min_x \| y - Dx \|_2^2 + \lambda \sum_i x_i, \text{ subject to } \forall_i x_i \geq 0. \quad (6.2)
$$

The non-negative constraint reduces the need for an absolute value over the entries of the coefficient vector $x$. One possible iterative technique is the following [60],

$$
x^{t+1} = x^t \ast (D^T y) ./ (D^T Dx^t + \lambda),
$$

where $\ast$ and $./$ represent entry-wise multiplication and division, respectively. Kreutz-Delgado and Murray showed a non-negative version of the FOCUSS algorithm [79], referred to as FOCUSS+. They proposed to project the results after each iteration onto the positivity constraints by setting to zero all negative elements.

A variation of the Orthogonal Matching Pursuit (OMP) for non-negative decomposition can also be easily derived. This includes selection of the dictionary atom that results the largest projection on the residual signal (without applying absolute value operation), followed by a non-negative least squares operation.
However, the performances of the OMP were found to be inferior than the latter methods, and therefore it was not used in our tests.

A thorough analysis concerning the linear programming solution of the convex problem,

$$\min_{x} \|x\|_1 \quad \text{subject to} \quad y = Dx, \quad x \geq 0,$$

(6.4)

was recently given by Donoho and Tanner [21]. They studied the connection between the true sparsest solution and the approximated one derived from the solution of (6.4). Considering the intrinsic properties of the dictionary $D$, and in particular, the convex hull of the point-set that contains the columns of $D$, conclusions regarding the equivalence between the two problems were drawn.

### 6.2.3 Design of Non-Negative Dictionaries - Prior Art

A simple method for non-negative matrix factorization that finds iteratively both the dictionary and the coefficient matrices was introduced by Lee and Seung in [69]. However, this method does not encourage the coefficients’ sparsity, and therefore is not designed for finding overcomplete dictionaries. In [68] they introduced their algorithm as a method for revealing the parts constructing the training signals, and presented their results working on a set of face images. The corresponding dictionary elements became localized, and each element contained different parts of the face. Hoyer [60, 61] developed an improvement for Lee and Seung’s algorithm, by enforcing sparsity constraints, therefore allowing the work with overcomplete dictionaries. He repeated the same tests with similar results.
6.2.4 Non-Negative K-SVD - NN-K-SVD

In order to adapt the K-SVD for producing non-negative dictionaries (and coefficient matrices) two slight changes should be done. In the sparse coding stage, an adequate pursuit method must be used, forcing non-negative coefficients, as described above. We preferred to use the iterative method presented in [60], also described in Equation (6.3), which is a variation of BP for non-negative decomposition. Furthermore, we added one change for this method, in order to allow finding a decomposition with a pre-specified number of coefficients, $L$.

After a couple of iterations are done, the indices of the $L$ largest coefficients are selected, and the data is approximated by those elements alone, using least-squares with non-negativity constraint on the coefficients. If we denote $\omega_L$ as the indices of the $L$ selected atoms, then we solved

$$
\min_x \|y - D^{\omega_L}x\| \quad s.t. \quad x \geq 0.
$$

(6.5)

using the Matlab’s function ‘lsqnonneg’, which uses the algorithm described in [67].

In the dictionary update stage, we must force the dictionary matrix to stay positive after each atom update. Our problem, in parallel to the one in Equation (4.10), is therefore

$$
\min_{d_k, x_k} \|E^{\omega_k} - d_k x_k^R\| \quad s.t. \quad d_k, x_k^R \geq 0.
$$

(6.6)

where $\omega_k$ is the set of data indices that uses the atom $d_k$ in their representation. As before, this problem reduces to finding the best positive rank-1 matrix that approximate the error matrix $E^{\omega_k}$, but adding a positivity constraint. This problem has the same complexity as the original SVD step, but in order to solve it we employ an iterative technique, as described in Figure 6.1, assuming $A = E^{\omega_k}$. The
initial solution for this method is chosen as the SVD solution, truncated to null
the negative entries. Note that the first singular vectors can both be multiplied
by \((-1)\) without changing the overall rank-1 approximation, and therefore both
options should be tested and compared. A full description is presented in Figure 6.1

\[
\begin{align*}
\text{Initialization: Set}\quad d(i) &= \begin{cases} 0 & \text{if } u_1(i) < 0 \\ u_1(i) & \text{otherwise} \end{cases}, \\
x(i) &= \begin{cases} 0 & \text{if } v_1(i) < 0 \\ v_1(i) & \text{otherwise} \end{cases},
\end{align*}
\]

where \(u_1\) and \(v_1\) are the first singular vectors of \(A\).

Repeat until convergence:

1. Set: \(d = \frac{Ax}{x^Tx}\)
   
   Project: \(d(i) = \begin{cases} 0 & \text{if } d(i) < 0 \\ d(i) & \text{otherwise} \end{cases}\)

2. Set: \(x = \frac{A^Td}{d^Td}\)
   
   Project: \(x(i) = \begin{cases} 0 & \text{if } x(i) < 0 \\ x(i) & \text{otherwise} \end{cases}\)

Figure 6.1: Finding a non-negative rank-1 approximation for a matrix \(A = dx^T\)

We often found that the true local minimum is only slightly different from
the initial solution supplied by the SVD projection to the non-negative space,
and therefore, we decided to skip the iterative method in cases where the initial
solution supply a sufficient reduction of the error. Notice that setting the negative
values in the error matrix to zero, and applying SVD, also ensures us positive
updated elements, but produces worse results.

At the end of this iterative procedure, the vector \(d_k\) should be normalized.
by dividing it by a scalar, as it construct a dictionary element, and $x^k$ should be multiplied by the same scalar. The full K-SVD variation for non-negative factorization, denoted as NN-K-SVD is presented in Figure 6.2.

### 6.2.5 A Synthetic Experiment

The following synthetic experiment was done with the NN-K-SVD. We manually generated 10 dictionary elements of size $8 \times 8$ pixels, containing the images of the 10 decimal digits. Each digit was then translated by 1 pixel up/down and left/right, constructing 9 possible configurations, resulting with a total of 90 dictionary elements. This dictionary is presented on the upper left side of Figure 6.3. A set of 3000 training signals was generated as random combinations of 5 such atoms, with random positive coefficients. The test was conducted twice, with no noise and with 15dB noise. Those 3000 training signals were used as input to the NN-K-SVD, which resulted with the positive dictionaries presented in the two upper right images (for the two tests) of Figure 6.3. The NN-K-SVD run was stopped after 60 and 100 iterations respectively. We also used the same data with Hoyer’s algorithm [60], which was stopped after 1500 iterations. The resulted dictionaries are presented in the second row of Figure 6.3. Information about each result is given in the Figure. Note that in this test the NN-K-SVD had an advantage because it used the exact number of coefficients, while Hoyer’s algorithm was executed as is with a sparsity factor of 0.8 (see [61]) on the coefficients.
**Initialization**: Set the non-negative random normalized dictionary matrix $D^{(0)} \in \mathbb{R}^{n \times K}$.

Repeat until convergence,

- **Sparse Coding Stage**: Use any pursuit algorithm for non-negative decomposition to compute $x_i$ for $i = 1, 2, \ldots, N$

  \[
  \min_x \left\{ \|y_i - Dx_i\|_2^2 \right\} \quad \text{subject to} \quad \|x\|_0 \leq T_0 \land \forall i \; x_i \geq 0.
  \]

- **Codebook Update Stage**: For $k = 1, 2, \ldots, K$
  * Define the set of examples that use $d_k$,
    \[ \omega_k = \{ i \mid 1 \leq i \leq N, \; x_i(k) \neq 0 \}. \]
  * Compute
    \[
    E_k = Y - \left( DX - d_k x^k \right),
    \]
  * Restrict $E_k$ by choosing only the columns corresponding to $\omega_k$, and obtain $E_k^{\omega_k}$.
  * Calculate $d_k$ and $x^k$ as described in Figure 6.1. Normalize $d_k$.

Figure 6.2: NN-K-SVD
Figure 6.3: On top, from left to right: True initial dictionary, K-SVD results after 60 iterations in the no-noise test, and after 100 iterations when the noise level was 15 SNR. On the bottom, from left to right: Hoyer’s algorithm results in the no-noise case after 600 and after 1500 iterations, and after 1500 iterations in the test with noise level of 15 SNR.
6.3 Shift-Invariant K-SVD (SI-K-SVD)

6.3.1 General

In this section we show a variation of the K-SVD algorithm for training a shift invariant dictionary \( D = \{d_i\}_{i=1}^{L} \in \mathbb{R}^{m \times L} \) for \( m \leq n \). Each signal \( y \) is represented as a sum of shifted atoms \( S \cdot d_i \), for \( S_i \) a matrix that inserts a vector of size \( m \) in the \( t \)'th location within a longer vector of size \( n \), assumed to be zero elsewhere. This means that the number of representing atoms is \( sL \), where \( s \) is the number of possible shifts of each atom.

6.3.2 Training Shift Invariant Dictionary - Prior Art

Several works dealt with the problem of training a shift invariant dictionary. In [4] a variation of the original algorithm proposed by Field and Olshausen [82] is proposed. There, a new update rule that takes into account all possible shifts is derived, with a slight change in the coefficient values, in order to preserve the assumed sparse distribution.

In a recent work of Engan et al. [42], variations of the MOD [39] algorithm for constrained dictionaries are presented by the name ‘ILS-DLA’ (for ‘Iterative LS-based Dictionary Learning Algorithms’). This work includes shift-invariant constraints (referred to there as an overlapping dictionary, with \( N = 1 \)), similar to the way we will use here. As in the MOD, the optimal constraint dictionary matrix is found, assuming fixed coefficients. In Section 2.3.3, as also in [1], it was shown that the dictionary update rule of ‘ILS-DLA’ is the expected outcome of the iterated update in the algorithm of [4].

A different approach was proposed in [63] by the name ‘MoTIF’ (for ‘Match-
ing of Time Invariant Filters’) for this exact purpose. There, an iterative and greedy method is suggested, in which in each step, a new atom is found. This atom is required to be best correlated to the training signals, while being the least correlated to the other already found atoms.

6.3.3  K-SVD for Shift-Invariant Dictionaries (SI-K-SVD)

In learning shift invariant dictionaries we train a dictionary $D \in \mathbb{R}^{m \times L}$ where $m < n$. Each signal is represented by a linear combination of shifted atoms. Let $S_i \in \mathbb{R}^{n \times m}$ be a shifting matrix with only $m$ nonzero entries which equal 1 in the semi-diagonal that starts in the $i$'th row,

$$S_i = \begin{cases} (i-1) \text{lines} \\ \text{(identity matrix)} \\ (n-m-i+1) \text{lines} \end{cases} \begin{bmatrix} 0 & \ldots & 0 \\ \vdots \\ 0 & \ldots & 0 \end{bmatrix}$$

and $s$ be the number of possible shifts. For the natural case of allowing all shifts from $i = 1$ to $i = n - m + 1$, we obtain $s = n - m + 1$. One could also allow cyclic positions getting $s = n$. Then, a representation of a signal $y$ using a linear combination of $T_0$ elements is expressed by,

$$y_i = \sum_{i=1}^{T_0} x_i S_{i} d_i.$$
The trained dictionary should then minimize,

\[
D = \arg \min_D \sum_{i=1}^{N} \left\| y_i - \sum_{j=1}^{T_0} x_{ij} S_{ij} d_{ij} \right\|_2^2 .
\]  

(6.9)

The **sparse coding** stage for a shift-invariant dictionary is not different from the regular sparse coding procedure, considering the full dictionary of size \(n \times (sL)\). However, the implementation can be made more efficient by taking into account the fact that the multiplication of the adjoint of the dictionary by a signal amounts to \(L\) convolutions. For further discussions on sparse coding algorithms for shift invariant dictionaries see [63, 4]. We note here that this sparse coding stage is problematic, and prevents the overall training algorithm to reach a successful solution, as will be described later.

In the **dictionary update** stage, the dictionary atoms are updated one after the other. When updating the atom \(d_k\), we consider the signals it represents as also its relevant shifts as fixed, and allow changing only the values of its entries and its coefficients. As in 4.2.2, Let \(\Omega_k\) be the set of all data indices that use \(d_k\) in their representation. we define the representation error matrix without the atom \(d_k\),

\[
E^k = Y^{\Omega_k} - \left\{ \sum_{l=1}^{T_0} x_{jl} S_{jl} d_{jl} \right\} .
\]  

(6.10)

The representation error matrix \(E^k \in \mathcal{R}^{n \times |\Omega_k|}\) should be approximated by setting new values for \(d_k\) and its coefficients, but without changing their support and shifts. Our objective is therefore written as

\[
\min_{d_k, \{x_{ij}, j \in \Omega_k, d_i = d_k\}} \left\| E^k - \left\{ \sum_{\{d_i = d_k\}} x_{ij} S_{ij} d_{ij} \right\} \right\|_2^2 .
\]  

(6.11)
Notice that this problem is different from the one we encountered before (Equation 4.10), because several of the entries in $E^k$ are not affected at all from $d_k$, while other entries may be approximated by a linear combination of several different entries in $d_k$, because of overlapping shifts. Therefore, a simple rank-1 approximation of the error matrix is not enough (see Figure 6.4).

For simplicity, we reduce our problem into approximating one column $e_i$ of $E^k$,

$$
\min_{x_1, x_2, \ldots, x_{T^k}, d_k} \left\| e_i - \sum_{j=1}^{T^k} x_{i_j} S_{i_j}^T d_k \right\|^2_2,
$$

(6.12)

where $T^k_i$ is the number of times the atom $d_k$ participate in the representation of $y_i$. Our objective is to minimize this error with respect to both $x_{i_j}$ and the atom $d_k$. The shifts $S_{i_j}$ are known and assumed fixed. As before, we iterate between the update of $x_{i_j}$ and $d_k$. Fixing the coefficients $x_{i_1}, x_{i_2}, \ldots, x_{i_{T^k}}$, we optimize the atom by setting

$$
d_k = \left( \sum_k \sum_j x_{i_k} x_{i_j} S_{i_k}^T S_{i_j} \right)^{-1} \sum_{j=1}^{T^k} x_{i_j} S_{i_j}^T e_i.
$$

(6.13)

Note that the matrix to invert is symmetric of size $m \times m$. The arguments for which the difference between $i_k$ and $i_j$ is greater or equals $m$ become zero. Then, we fix the value of $d_k$ and set new values for the coefficients, one at a time by

$$
x_{i_j} = \frac{d_k^T S_{i_j}^T e_i - \sum_{k \neq j} x_{i_k} d_k^T S_{i_j} S_{i_k} d_k}{d_k^T d_k}.
$$

(6.14)

For solving for all vectors in $E^k$,

$$
\min_{x_1, x_2, \ldots, x_M, d_k} \left\| E^k - \left\{ \sum_{i=1}^{T^k} x_{i_j} S_{i_j} d_k \right\}_{j \in \Omega_k} \right\|^2_2,
$$

(6.15)
The sum of the above signals

Figure 6.4: Illustration for approximation of a vector by a linear combination of 3 shifted versions of the atom \( \mathbf{d} \).
where \( M = \sum_{i \in \Omega_k} T_i^k \), we concatenate the columns one after the other, creating one long error vector \( \bar{e} \in \mathcal{R}^{n|\Omega_k|} \), while a respective change in the shifting matrices is done, so as to reduce the problem to the one in (6.12).

### 6.3.4 SI-K-SVD - experiments

For validation of the algorithm, we synthetically created 2000 signals of dimension 20 using linear combinations of 3 shifted atoms from a dictionary \( D \in \mathcal{R}^{18 \times 45} \) (without allowing cyclic shifts, so that the number of possible shifts was 3). Thus, the overall effective dictionary is of size 20 \( \times \) 135. The selected indices and their shifts are randomly chosen. White Gaussian noise of several power settings was added in order to generate SNR values of 10dB, 20dB, 30dB and without any noise. The SI-K-SVD algorithm was executed for 80 iterations, while its initial choice for \( D \) was the first 18 entries in the first 45 signals. For comparison, we implemented the ILS-DLA algorithm for shift invariant dictionaries [42], which was used in a similar manner to the SI-K-SVD (same pursuit method, same number of iterations, same initialization). We also tested the original K-SVD algorithm on the same signals, for the same number of iterations, trying to extract 135 dictionary atoms that correspond to the 45 atoms, in all shifts. For all methods, the learned dictionary is compared to the initial true dictionary, using a similar procedure as in Section 4.2.5 and in [1]. Each test, in each method and in each noise level was executed 5 times. The average detection rate was calculated, and is presented in Figure 6.5.

As can be seen, the advantage of the SI-K-SVD over the ISL-DLA is not big. However, the advantage over applying a ‘regular’ scheme is clear. Moreover, we can see that the level of noise does not effect the overall results of the algorithms.
Figure 6.5: Each symbol represents the average detection rate of 5 tests. The tested methods were SI-K-SVD, ISL-DLA for shift invariant dictionaries, and the original K-SVD method (without shift invariance).
that force shift-invariance, while it highly effects the 'regular' method. This proves the need for a shift-invariant variation, when the underlying dictionary is known to hold this property.

We also implemented the SI-K-SVD on patches from real images. The size of each patch was $10 \times 10$, and each dictionary elements was set to be $4 \times 4$. A fixed number of 5 shifted atoms was set to represent each patch, and the number of different atoms was set to 20 (resulting $D \in \mathbb{R}^{16 \times 20}$). The resulted dictionary atoms are presented in the center of each block in Figure 6.6.

![Figure 6.6: Detected atoms found by the SI-KSVD algorithm executed on 10 x 10 patches from natural images. The found atoms of size 4 x 4 are located in the center of each block, for visualization purposes.](image)

6.3.5 Summary

We introduced a variation of the K-SVD algorithm for learning shift invariant dictionaries. This algorithm ensures a reduction in the representation error in each iteration in the ‘dictionary update’ stage. However, its performances are only partially successful.
We believe that the main problem of the SI-K-SVD algorithm, as also the other training algorithms presented in this section, is the problematic sparse coding stage. Using a prior definition, the *mutual coherence* of a shift invariant dictionary is relatively large, preventing the pursuit algorithms to reach a successful solution. We assume an adequate solution to the sparse coding problem will directly derive more successful training abilities.

### 6.4 Linear Constraints

We now turn to a more general constraint, in which each atom in the dictionary $\mathbf{D}$ is known to be a product of the form,

$$\forall i, \quad \mathbf{d}_i = \mathbf{R}^i \tilde{\mathbf{d}}_i,$$

(6.16)

where $\mathbf{R}^i \in \mathbb{R}^{n \times m}$, is a known matrix of rank $r$, $\tilde{\mathbf{d}}_i \in \mathbb{R}^m$ and $n \geq m \geq r$. Note that if $n = m = r$ there is no constraint on $\mathbf{d}_i$ and the problem simplifies to the original K-SVD setting.

This constraint is different from the linear constraint suggested by Engan et al. [42], which enforces $\mathbf{A} \tilde{\mathbf{D}} = \mathbf{b}$, where $\tilde{\mathbf{D}}$ is an arrangement of the dictionary $\mathbf{D}$ as a vector. The latter is a linear constraint on the dictionary $\mathbf{D}$, while our constraint forces each $\mathbf{d}_i$ to be a linear transformation of another, un-constrained, dictionary atom $\tilde{\mathbf{d}}_i$. Some structures can be enforced by the two constraints (like symmetry, identical entries, zero entries, etc.). Other structures, such as those that relate to the connection between different atoms, can be more easily enforced using the global linear constraint in [42]. Structural constraints that are functions of other matrices, such as forcing an atom to be an enlarged version of another one, can be more easily applied by our approach. An example for such a requirement is presented in Section 6.4.2, where we discuss a multiscale version of the dictionary.
6.4.1 Linear Constraint K-SVD (LC-K-SVD)

As before, the sparse coding stage of each iteration is not changed, as also the definition of $E^k$ (Equation 4.10), since we can use the effective dictionary with the given atoms and apply pursuit techniques directly. However, in the dictionary update stage the new value for the atom $d_k$ should obey $d_k = R^k \hat{d}_k$. Therefore we solve,

$$\min_{d_k, x_{\Omega_k}^k} \| E^k - d_k x_{\Omega_k}^k \|_F^2 \quad s.t. \quad d_k = R^k \hat{d}_k,$$

or alternatively,

$$\min_{d_k, x_{\Omega_k}^k} \| E^k - R^k \hat{d}_k x_{\Omega_k}^k \|_F^2.$$

Let the SVD factorization of $R$ be\(^1\),

$$R^k = U_R \Sigma_R V_R^T,$$

where $U_R \in \mathcal{R}^{n \times n}$, $\Sigma_R \in \mathcal{R}^{n \times m}$ with $r$ nonzero entries on the main diagonal $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$, and $V_R \in \mathcal{R}^{m \times m}$. Then,

$$\| E^k - R^k \hat{d}_k x_{\Omega_k}^k \|_F^2 = \| E^k - U_R \Sigma_R V_R^T \hat{d}_k x_{\Omega_k}^k \|_F^2,$$

$$= \| U_R^T E^k - \Sigma_R V_R^T \hat{d}_k x_{\Omega_k}^k \|_F^2.$$ 

Let $F = U_R^T E^k = \begin{bmatrix} F_1 \ F_2 \end{bmatrix}$ such that $F_1 \in \mathcal{R}^{r \times [\Omega_k]}$, and $F_2 \in \mathcal{R}^{(n-r) \times [\Omega_k]}$. Let us also denote $w = V_R^T \hat{d}_k = \begin{bmatrix} w_1 \ w_2 \end{bmatrix}$ such that $w_1 \in \mathcal{R}^{r \times 1}$, and $w_2 \in \mathcal{R}^{(m-r) \times 1}$.

\(^1\) The following development is based on a personal communication with Prof. Shmuel Friedland.
Finally, let $\Sigma_R \in \mathbb{R}^{r \times r}$ be a diagonal matrix with $\sigma_1, \sigma_2, \ldots, \sigma_r$ on its diagonal, then

$$
\left\| E^k - R^k d_k x^k_{\Omega_k} \right\|_F^2 = \left\| \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} - \Sigma_R \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} x^k_{\Omega_k} \right\|_F^2 \quad (6.21)
$$

$$
= \left\| F_1 - \Sigma_R^r w_1 x^k_{\Omega_k} \right\|_F^2.
$$

Solving (6.21) requires finding the rank-1 approximation of $F_1$. If we use, again, the SVD factorization of the matrix $F_1 = U_F \Lambda_F V_F^T$, then, in parallel to the rank-one approximation described in (4.2.2)

$$
w_1 = \Sigma_R^{-1} U_F^{(1)},
$$

$$
x^k_{\Omega_k} = \Lambda_F(1,1) \cdot V_F^{(1)T}.
$$

For finding $d_k$ we solve $V_F^{(1,2,\ldots,r)^T} d_k = w_1$, which is a linear system of $r$ equations with $m$ variables, and as $r \leq m$ at least one solution exists (if $r < m$ anyone of the infinite number of solutions may be chosen). A final normalization of the atom $d_k = R^k d_k$, in addition to a respective change in the coefficients $x^k_{\Omega_k}$ might also be needed, if we require normalized elements of $D$.

6.4.2 Example - Multiscale Representation

6.4.2.1 Multiscale representation description

In the attempt to represent images using trained dictionaries so far, only small patches are able to be handled. Training a dictionary for full (even relatively small) $\sqrt{N} \times \sqrt{N}$ images is impractical – the dimension of such a dictionary atom would be $N$, and the number of them should be at least $N$ for completeness.
Training such a dictionary requires too much computational effort, as also too much memory. It would also lack the flexibility in working with various size images.

Therefore, as we have already suggested in Section 4.2.4, the solution is to train dictionaries for multiscale representation. Here we suggest to train dictionaries for representing a variation of the Laplacian Pyramid. Representation of images using a multiscale approach was already suggested and applied in various transforms, such as curvelets [20] and Countorlets [18]. Both these transforms are redundant, and encourage sparsity. However, they are based on fixed mathematical models, and do not arise from the signals they serve by training, which might be a disadvantage, especially when treating limited families of signals. In this section we suggest a way to combine the ideas of multiscale representation together with the idea of trained dictionaries, leaning on the above proposed paradigm of constrained dictionaries.

Olshausen and Sallee [92] presented an approach for learning a multiscale representation for images. In their algorithm, several mother wavelets function are trained, and the full image is then represented by a combination of spatially localized such functions with possible shifts and 2:1 scale. The learned functions were trained to represent the steerable pyramid, and their results were compared to the steerable transform with a classic wavelet, achieving slightly better quality in representation of images for the same level of sparsity, and better denoising results (however, worse than those presented in Section 5).

Our representation scheme for an image of size $\sqrt{N} \times \sqrt{N}$, where $N = n \cdot 4^m$ for some $m \in \mathbb{Z}$, is the following. We first refer to the coarsest version of the image of size $\sqrt{n} \times \sqrt{n}$ that contains the lowest frequencies. This level is referred to here
as the zero-level, or $I^0 = S^{m-0} \bar{I}$, where $S^{i-j} \in \mathcal{R}^{(4^i \cdot n) \times (4^i \cdot n)}$ is a resizing operator from images in level $i$ to images in level $j$, and $\bar{I}$ is an arrangement of the elements in $I$ (the given original image) as a vector. $I^0$ is represented by one dictionary $D^0 \in \mathcal{R}^{n \times L_1}$, multiplied by a sparse coefficient vector $x^0$. Then, we represent the differences between the enlarged version of the zero level representation $I^0$, $S^{0-1}D^0x^0$ and the reduced version of $I$, $R^1 = S^{m-1} \bar{I} - S^{0-1}D^0x^0$, $R^1 \in \mathcal{R}^{4n}$, so we use here 4 non-intersecting blocks, each of size $n$, represented by another dictionary $D^1 \in \mathcal{R}^{n \times L_2}$.

The next level to represent is computed similarly, as the difference between the enlarged version of what was already represented, and the reduced version of the image to represent,

$$R^2 = S^{m-2} \bar{I} - \left( S^{0-2}(D^0x^0) + S^{1-2}(\sum_{b=1}^{4} B^1_b D^1 x^1_b) \right),$$

(6.23)

where $B^1_j$ is an operator that places the $\sqrt{n} \times \sqrt{n}$ block into the $j$’th location of a $4^j \cdot n$ vector. In General, the $k$’th represented level will be,

$$R^k = S^{m-k} \bar{I} - \left( \sum_{l=0}^{k-1} S^{l-k} \sum_{b=1}^{4^l} B^l_b D^l x^l_b \right).$$

(6.24)

The dictionaries to be trained are $\Delta = \{D^0, D^1, ..., D^m\}$.

This representation scheme is presented in Figure 6.7, where $\uparrow L_i$ or $\downarrow L_i$ represent resizing (enlarge or reduce, respectively) to level $i$, $A(D^i)$ and $S(D^i)$ represent analysis and synthesis, respectively, using the $i$’th dictionary.

Actually, we may imagine the representation of $\bar{I}$ by a single very large dictionary matrix $D^M$ (for multiscale) of dimension $N \times \left( \sum_{i=1}^{m} 4^i L_i \right)$. The first $L_1$ atoms of $D^M$ are the atoms of $D^0$, enlarged to the maximal level, and therefore $D^M_i = S^{0-m} \cdot d^0_i$, for $1 \leq i \leq L_0$. Similarly, the next $4^1 \cdot L_1$ atoms of $D^M$ are enlarged
Figure 6.7: MultiScale representation scheme. $\downarrow L_i$ or $\uparrow L_i$ represent enlarge or reduce size to level $i$, $A(D^i)$ and $S(D^i)$ represent analysis and synthesis, respectively, using the $i$'th dictionary.
versions of the atoms of $\mathbf{D}^1$, placed in the relevant block, $\mathbf{D}_i^M = \mathbf{S}^{1-m} \mathbf{B}_i^l \mathbf{d}_j^l$, for $L_0 + 1 \leq i \leq L_0 + 4L_1$, and $b = \lfloor (i - 1 - L_0)/L_1 \rfloor + 1, j = ((i - L_0 - 1)\%L_1) + 1$ (modulus operator), and in general,

$$
\mathbf{D}_i^M = \mathbf{S}_k^i \cdot \mathbf{B}_b^k \mathbf{d}_j^k, \quad \text{for} \quad \sum_{l=0}^{k-1} 4^l L_l + 1 \leq i \leq \sum_{l=0}^{k} 4^l L_l \\
b = \lfloor (i - 1 - \sum_{l=0}^{k-1} 4^l L_l)/L_k \rfloor + 1, \\
j = \left( (i - \sum_{l=0}^{k-1} 4^l L_l - 1)\%L_k \right) + 1
$$

To summarize thus far, $\mathbf{D}^M$ is a huge dictionary that contains shifted and enlarged versions of the atoms in $\Delta$. To illustrate the size of such a dictionary, we consider $n = 8^2$ and $N = 256^2$ ($m = 5$), and redundancy $\mathbf{D}^l \in \mathbb{R}^{64 \times 256}$ for $l = 0, 1, \ldots, 5$, and get a dictionary $\mathbf{D}^M$ of size $65536 \times 349440$ for representation of images of size $256 \times 256$. However, rather than being a general dictionary with $\sim 2.3e^{10}$ free parameters, it is entirely defined by the 6 small dictionaries that construct it, containing together 70656 parameters. Of course, this dictionary cannot and will not ever be explicitly stored. Nevertheless, we remember its structure, as it will guide us when training the smaller dictionaries that construct it.

### 6.4.2.2 Sparse Representation of Images in a Multiscale scheme

A multiscale and sparse representation of an image as described above, each $\sqrt{n} \times \sqrt{n}$ block in each representation level $\mathbf{R}^l$ should be represented using the related dictionary $\mathbf{D}^l$. Each representation level may suffer errors, which then
propagate to the next level (see 6.24). Two extreme approaches can be adopted,

- Represent each level $R^i$ without any error.

- Not to represent the coarser levels $0, 1, \ldots, m-1$ ($x_j^i = 0$ for $i = 1, 2, \ldots, m-1$ and for all $j$), and represent only the blocks in $R^m$, which is practically a single level representation of $I$.

In an effective multiscale scheme, both the above extreme cases should be avoided. Partial representation of each layer and a proper propagation of the error should be practiced. However, such a process raises the question of how accurate should every layer be represented – a question of allocation of effort that we leave open in this work. A possible way to deal with this problem was suggested in [107], where a dynamic programming method is applied in order to best balance between the number of bits used and the distortion incurred. In this work we chose to use a fixed and pre-defined number of coefficient $T^l_0$ for each level $l$, in parallel to (4.4). Thus we should solve

\[
\min_{\{x_{ij}\}, 0 \leq i \leq m, 1 \leq b \leq 4^l} \sum_{l=0}^m \left\|R^l - \sum_{b=1}^{4^l} B^l_b D^l_b x^l_b \right\|_2^2 \quad \text{s.t.} \quad \forall_{l,b} \|x^l_b\|_0 \leq T^l_0.
\]  

(6.26)

This expression requires an adequate representation of each layer by representation of the distinct blocks, restricting the number of non-zero entries in each block representation from level $l$ to be less or equal $T^l_0$.

**6.4.2.3 Training Dictionaries for a Multiscale Representation**

For training the dictionaries in $\Delta$ we note two facts,

- Considering (6.25), each atom in $D^M$ is a product of some known matrix $S^{k-m} B^k_b \in \mathbb{R}^{N \times n}$ and an arbitrary atom $d_i \in D^k$ of length $n$. This leads to using the K-SVD variation for linear constraint dictionaries.
• Each atom $d_i \in D^k$, for $0 \leq k \leq m$ appears $4^k$ times in $D^M$, in different, but non-overlapping, locations. This reminds us of the shift-invariant property, where each column in the dictionary can appear in several different locations in a representing atom.

All this leads to some combination between the two latter K-SVD variations we described. In updating the atom $d_i \in D^k$, for some $0 \leq k \leq m$, The main idea is to calculate the representation error matrix $E^{ki}$. We do so in two stages. First, we calculated $E^k$, which is the representation error matrix without all atoms in $D^k$ (current level dictionary),

$$E^k = \tilde{I} - \sum_{l=0,l\neq k}^{m} \left( S^{l-m} \sum_{b=1}^{4^l} B_b^l D^l x^l \right). \quad (6.27)$$

$E^k \in \mathcal{R}^N$ holds the representation error excluding level $k$, which should be represented by $D^k$. For updating $d_i \in D^k$, we then add the representation by all other atoms in $D^k$, excluding $d_i$,

$$E^{k'} = E^k + S^{k-m} \sum_{b=1}^{4^k} \sum_{j=1,j\neq i}^{L_b} B_b^j d^j x^j[j]. \quad (6.28)$$

The vector $E^{k'}$ is the one to be minimized by setting new values for the atom $d_i^k$ and its coefficients. We collect all $(4^k \cdot n)$ blocks from $E^{k'}$ for which $d_i^k$ takes part in their representation, into columns in one matrix $E^{k'} \in \mathcal{R}^{(4^k \cdot n) \times |\Omega_n|}$. Doing so is equivalent to collecting all parts of the error matrix that overlap the nonzero entries in the updated atom in the shift invariant dictionary. As the blocks are non-overlapping, there could not be an entry in $E^{k'}$ that is affected twice by the atom $d_i^k$. This process also neutralizes the effect of the shifting matrix $B_b^k$. The new problem is then,
\[
\min \left\| E^k - S^{0-k} d_i^k x_i^k \right\|_2^2 (6.29)
\]

where \( x_i^k \) is a row vector that contains the nonzero values in the \( i \)'th row of all vectors \( x_b^k \), for \( b = 1, 2, \ldots, 4^k \). We assume that \( S^{0-k} \) is full rank, so according to (6.21), only the \( n \) first left singular vectors are relevant for the rank-one approximation. Surprisingly, those \( n \) columns are similar to the size-reduction operator \( S^{k-0} \). This means that the columns of \( E^k \) should be reduced to size \( n \), and then a rank-one approximation should be found.

If we assume that enlarging an image and then reducing its size back to the original size results with the initial image (or formally, \( S^{i-j} S^{j-i} = I \) when \( j < i \)), then we conclude that the resizing in Equation (6.27) can be done to level \( k \), while no enlargement at all is needed in (6.28).

### 6.4.2.4 Experiments

Synthetic tests to find the multiscale dictionary underlying the data were first applied on 1000 synthetic signals. Each signal was of size \( 12 \times 12 \), while the basic block size was set to \( 3 \times 3 \) (which implied 3 levels of representation). Each dictionary in each level consisted of 9 elements (complete dictionary), which implies \( 1 + 4 + 16 = 21 \) different underlying dictionaries, and \( 21 \cdot 9 = 189 \) atoms. Fixed numbers of coefficients for representation of a block were assigned to each level – 2, 2, and 1 to the coarsest, middle, and finest levels respectively. Therefore, each representation of a \( 12 \times 12 \) signal includes under this setting \( 2 + 2 \cdot 4 + 1 \cdot 16 = 26 \) coefficients. All executions were performed for 30 iterations. The detection rate for each level was measured in the same method as in 4.2.5. An average detection rates of 5 different experiment for the 3 levels and for 4 different noise levels are
presented in Table 6.1.

The initialization of the learned dictionary was found to be crucial for the success of the algorithm. We initialized the dictionaries by the results of another single-scale learning process, that was executed on blocks of images, extracted from the training set. The blocks that were used for learning the initial dictionary $D^0$ were $y^0_i = S^{m-0}I_i(:,)$, where $I_i$ is the $i$’th training signal, arranged as a column vector. For extraction of the input signals for learning the initialization of the other dictionaries, we assumed perfect representation of the layers, and set,$y^l_i = B^l_b(S^{m-l}I_l(:,)) - S^{(l-1)-l}S^{m-(l-1)}I_i(:,),$ \hspace{1cm} (6.30)

where $b$ is some random block chosen from the $4^l$ blocks in the image. That is, the blocks that were used for training the initial $D^l$ dictionary were taken from a differences image, resulted by extracting the enlarged version of the original, reduced to level $l - 1$ ($S^{l(l-1)-l}S^{m-(l-1)}I_i(:,)$) from the original image, reduced to level $l$ ($S^{m-l}I_l(:,)$).

<table>
<thead>
<tr>
<th>level</th>
<th>No noise</th>
<th>30dB</th>
<th>20dB</th>
<th>10dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>71.11</td>
<td>60</td>
<td>57.78</td>
<td>57.78</td>
</tr>
<tr>
<td>1</td>
<td>66.67</td>
<td>75.56</td>
<td>53.33</td>
<td>62.22</td>
</tr>
<tr>
<td>2</td>
<td>71.11</td>
<td>75.56</td>
<td>53.33</td>
<td>62.22</td>
</tr>
</tbody>
</table>

Table 6.1: Average detection rates (in percentage) for the multiscale synthetic experiments. Notice the number of atoms are 9, 36 and 144 for the zero (coarsest), first and second level, respectively. Each experiment was performed 5 times.

This initialization was found successful, and enabled performing only a few iterations until convergence. A typical progress of one of the executions is presented in Figure 6.8. As can be seen, the initial dictionaries were quite successful before even starting the multiscale K-SVD process.
Figure 6.8: Typical execution of MS-KSVD (30dB). The graph presents the detection ratios of each level (the zero level is the coarsest one) for the first 6 iterations.
In trying to process real images, we executed the K-SVD on real $64 \times 64$ images (basic block size: $8 \times 8$, 4 levels). The training set included front-view aligned images, 4 of which can be seen in Figure 6.9. The mean value of each image was reduced before training. The 4 extracted dictionaries of size $64 \times 128$, were initialized by the single-level K-SVD learning process, as described above. The resulted dictionaries after 80 iterations are presented in Figure 6.10. We can see that the finer the level of representation, the sharper and clearer are the edges.

![Figure 6.9: Samples from the 64 x 64 face images used for training the multiscale dictionaries.](image)

### 6.5 Image Signature Dictionary

#### 6.5.1 Introduction

Let $\mathbf{D} \in \mathcal{R}^{N \times M}$ be an ‘Image Signature Dictionary’ (ISD). Each signal block $\mathbf{y} \in \mathcal{R}^{n \times m}$ can be represented by a linear combination of blocks (atoms) of the same size, taken from arbitrary locations in $\mathbf{D}$. If $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{D}}$ are arrangements of $\mathbf{y}$ and $\mathbf{D}$ as columns, then,

$$
\tilde{\mathbf{y}} \approx \sum_{i} \alpha_i \mathbf{S}_i \mathbf{D} \quad (6.31)
$$
Figure 6.10: Multiscale dictionaries extracted by the K-SVD algorithm, processed on real face images. The most left dictionary is the coarsest.
where $S_i \in \mathbb{R}^{nm \times NM}$ is the block selection matrix, having all zero entries except \( nm \) entries, one in each line, that equals 1. We shall assume that for all sizes \((n, m)\), \( N \cdot M \) patches from $D$ of that size are available because we allow cyclic transformation (see Figure 6.11).

The redundancy factor of $D$ is $\frac{NM}{nm}$ for each block size \((n, m)\). Yet, it requires very small space. One might say that this dictionary atoms are highly constrained, as each atom equals 4 other ones in almost all entries except of one row or column, and therefore its true redundancy is much smaller. However, such overlap between the atoms is natural for representing signals, as it is tightly coupled with the idea of shift-invariance, although in a different flavor from the one already discussed above. In what follows we shall describe how such dictionary is learned and used.

6.5.2 Training an Image Signature Dictionary

Given a set of training patches $y_i \in \mathbb{R}^{n \times m}$, we search for an ISD $D \in \mathbb{R}^{N \times M}$ that can lead to the best representation of all those patches. For simplicity we currently assume each patch is represented by a fixed number of atoms ($L$).

$$
\min_D \sum_i \left\| \tilde{y}_i - \sum_{j=1}^{L} \alpha_{ij} S_{ij} \tilde{D} \right\|_2^2
$$

We follow previously reported algorithms for training dictionaries and adopt the two stages iterated method - sparse coding and dictionary update. In the sparse coding stage, the coefficients’ supports and values are found, assuming a known and fixed dictionary. Then, the representation is assumed fixed and the dictionary is found to best minimize Equation (6.32).
Figure 6.11: Illustration of several atoms from in an ISD.
6.5.2.1 Sparse Coding

In representation of signals under such a scheme we should find the optimal \( \alpha_i \) and \( S_i \) that minimize the expression in (6.31). The simplest way to do so is to extract from \( D \) all possible atoms, and use one of the pursuit methods described in Section 2.1. However, the special structure of this dictionary can be used to find a more efficient pursuit technique. For example, all projections between a signal and all dictionary atoms can be computed by only one inner product between the Fourier transforms of the signal and the ISD, exploiting the equivalence between a convolution in the space domain and inner product in the frequency domain [85]. While a straight forward projection of a signal onto all possible atoms is done in \( O(nm \cdot NM) \), The same result can be achieved in \( O(NM \log(MN)) \), including forward and inverse Fourier transform of the signal.

6.5.2.2 Dictionary Update Stage

Each pixel in \( D \) is common to \( nm \) different atoms, and therefore affected from up to \( nm \) coefficients. Updating each pixel separately together with its relevant coefficients, as suggested by the K-SVD algorithm seems hopeless. We therefore follow the MOD algorithm and derive Equation (6.32) in respect to \( D \),

\[
\sum_i \sum_{j=1}^{L} \alpha_{ij} S^T_{ij} \tilde{y}_i - \sum_i \sum_{j=1}^{L} \sum_{k=1}^{L} \alpha_{ij} \alpha_{i,k} S^T_{ik} S_{ij} \tilde{D} = 0, \quad (6.33)
\]

implying the following update rule

\[
\tilde{D} = \left( \sum_i \sum_{j=1}^{L} \sum_{k=1}^{L} \alpha_{ij} \alpha_{i,k} S^T_{ik} S_{ij} \right)^{-1} \sum_i \sum_{j=1}^{L} \alpha_{ij} S^T_{ij} \tilde{y}_i. \quad (6.34)
\]

Each such update is promised to reduce the overall representation error, and if we assume the pursuit stage is successful, the overall algorithm must converge.
6.5.3 Experiments

We tested the applicability of such a dictionary for denoising of images. We adopt a similar method to the one suggested in Chapter 5. First, just as described there, we trained a global dictionary on patches of real images which do not belong to the test set. The reconstructed version is an average image of the represented overlapping blocks. Here, a dictionary of size $100 \times 100$ was trained, using the same training set that was used for training the global dictionary in Section 5.4, and is presented in Figure 6.12. Its denoising results are very similar (±0.1dB) to those reported in Table 5.1 for the global trained dictionary.

Next, the image signature dictionary is trained on the overlapping patches from the noisy image, and the reconstructed version is an average image of the represented blocks. The denoising results we received when we worked with a $(75 \times 75)$ size dictionary and $(8, 8)$ blocks are almost identical to those reported in Table 5.1. Several Trained dictionaries are presented in Figure 6.13, and the processed (clean) images are presented in Figure 6.14.

We also examined the shift-invariance property of the image signature dictionary. Starting with an image of size $160 \times 160$ (a portion of the image ’House’), we represent all its distinct $8 \times 8$ blocks (total of 400) using the global trained ISD (Figure 6.12). We then shifted the selected $160 \times 160$ image in $(x, y)$ pixels (meaning $x$ pixels down, and $y$ pixels left), and examined the quality of representation by shifting the used dictionary atoms respectively (or equivalently, changing the index of the non-zero coefficients). Then, we allowed each block to drop one atom (whose coefficient’s absolute value is minimal), and choose another atom (only one) instead. The results of 3 such shifts are presented in Figure 6.15.
Global Image Signature Dictionary

Figure 6.12: Image signature dictionary of size $100 \times 100$, trained on patches from real images.
Figure 6.13: Trained image signature dictionaries for several images. For each of these signatures we describe which image it is related to, the size of the signature, and the additive noise power that affected it.

Figure 6.14: Sample images we used for the various tests. From left to right: Barbara, House and Peppers.
Figure 6.15: Examining the shift-invariance property of the ISD.
Chapter 7

Steps Towards a Geometrical Study of \textit{Sparseland}

This entire work is based on the core assumption that interesting families of signals (such as images) can be described as emerging from the \textit{Sparseland} model. It is true? how can we assess such an assumption?

In this section we confront some theoretical questions concerning the representation abilities of ‘sparse coding’ schemes, and the ‘\textit{Sparseland}’ model in general. We use expressions and definitions from computational geometry and describe the structure of the set of signals that can be sparsely approximated by some dictionary. Also, given a specific dictionary, we set bounds on the size of the set of signals that can be represented. Therefore, we supply tools that can help in approximating the representation abilities of given dictionaries. We should note, though, that the proposed study served in the chapter is a very partial one, and further work is required to complete it.

7.1 Introduction and Motivation

The \textit{Sparseland} model, described in Section 1.3, is one of the main motives of this thesis. According to this model, a family of signals can be sparsely represented by some dictionary, and in ‘sparsely’, we mean that the number of
representing atoms is smaller, or even substantially smaller, than the dimension of each represented signal. However, the number of signals that can be \textit{exactly} represented by a linear combination of $L$ atoms from $D$, for $L < n$ and a finite $D$, is of measure zero in $\mathbb{R}^n$. Each $L$ atoms span a subspace of dimension $L$ in the $n$-dimensional space, which volume is negligible related to the whole space. Even multiplying this ratio by $\binom{K}{L}$, the number of possible subspaces, does not change this fact. Therefore, on what exactly the \textit{Sparseland} model relies on?

Let us start with the fact that we often work with digital signals, that bear some kind of quantization effects. In addition, Approximation is often allowed, representing only a close version of the initial signals $y$. This allowed representation error $\epsilon = \|\hat{y} - y\|_2$ widens the possible represented subspace, and gives a non-zero volume to the above subspaces, which can be measured. In this section we check the representation ability of sparse coding, by measuring those relative volumes of represented subspaces. We first check the ratio of all signals that can be represented by $L$ coefficients out of all signals in dimension $n$. We then discuss the representation abilities of a given dictionary, and provide lower and upper bounds for the relative represented space.

When designing a dictionary for a specific set of signals we require sparse representation of each element in this set. However, in many cases this property is not enough. We would also like the designed dictionary not to represent sparsely other signals, which do not belong to this set. Such a requirement makes this dictionary a prior that can serve well in inverse problems (such as the dictionary-prior described in Section 5). The bounds provided here are important to validate that only a small fraction of the signal space is covered by the \textit{Sparseland} model, in order to make sure that this prior is indeed discriminative.
7.2 Representation Abilities of $L$-atoms

In discussing the relative volume of all represented signals we will consider only the normalized signals, which lie on an $n$-dimensional hyper-sphere. When we represent signals using linear combination of dictionary atoms, a signal $y$ can be represented by the same atoms as its normalized version, and under the same relative error. Therefore, this kind of constraint does not limit our discussion.

Each set of $L$ independent atoms spans a subspace of order $L$. This subspace intersects with the $n$-dimensional hyper-sphere by another hyper-sphere of dimension $L$. For example, lets consider a surface that passes through the origin and intersects with the 3-dimensional unit ball whose center is the origin. This creates a 2-dimensional ball, i.e. a circle, of the same radius (similar to the equator drown on the globe).

We also allow a representation error which $\ell^2$ norm is no greater than $\epsilon$, that is:

$$\|y - Dx\|_2 \leq \epsilon.$$  \hspace{1cm} (7.1)

This allowed error widens the intersection area and its effect is just like a dilation operation with an $n$-dimensional ball of radius $\epsilon$. The result of such a dilation is an area that includes signals which are outside the $n$-dimensional hyper-sphere, and therefore should be intersected again with the unit $n$-dimensional hyper-sphere.

We go back to our simple example. The allowed error causes the intersection circle to become a torus, and after the intersection with the initial 3-dimensional hyper-sphere we obtain a strip on the ball (similar to the area covered by a few degrees of latitude northern and southern the equator). If the allowed error is small enough, the strip is very close to being a cylinder. In this section we will
calculate the surface area of this generalized cylinder (in all possible dimensions) and divide this by the surface area of the whole $n$ dimensional hyper-sphere, in order to get the ratio of represented signals out of all signals.

We will begin with basic geometrical facts. The surface area of a $d$-dimensional hyper-sphere with radius $r$ is given by

$$S_d(r) = \frac{2\pi^{d/2} r^{d-1}}{\Gamma(d/2)}, \quad (7.2)$$

where $\Gamma$ is a function defined by $\Gamma(x) = (x-1)\Gamma(x-1)$, $\Gamma(1) = 1$ and $\Gamma(1/2) = \sqrt{\pi}$. The volume of this hyper-sphere is

$$V_d(r) = \frac{\pi^{d/2} r^d}{\Gamma(d/2 + 1)}, \quad (7.3)$$

with the same definition for $\Gamma$.

The surface area of the intersection between the $L$-dimensional sub-space and the $n$-dimensional sphere is $S_L(1) = \frac{2\pi^{L/2} r^{L-1}}{\Gamma(L/2)}$. In order to calculate the full represented volume, we multiply the surface area of a $L$ dimensional sphere by the volume of an $(n - L)$ dimensional sphere with radius $\epsilon$, which represents the allowed deviation in the $n - L$ other dimensions. This multiplication gives us an approximation, actually an upper bound, to the true covered area. As we will see next, this approximation tends to be almost exact when the error is sufficiently small. The desired ratio, denoted as $r(n, L, \epsilon)$ is then:

$$r(n, L, \epsilon) = \frac{S_L(1) \cdot V_{n-L}(\epsilon)}{S_n(1)}. \quad (7.4)$$

For better understanding this issue, and the type of approximation we are using, we give some examples, in low dimensions.

When $n = 2$ and $L = 1$, we work in a two-dimensional world, in which all normalized signals construct a unit circle. Our represented signals lie on lines
that go through the origin (sub-spaces of dimension 1). The intersection of one line with the two-dimensional circle is two points, and allowing an error which energy is no larger than $\epsilon$ - derives a deviation from this point of no more than $\epsilon$ (see left side of Figure 7.1, where only one intersection point is presented). We consider only the normalized signals in this circle (the bold sector in the figure). The bounded length of this sector on the perimeter, denoted as $l(\epsilon)$, divided by the whole perimeter of the circle is the desired answer. What is the exact size of $l(\epsilon)$? It is relatively simple to answer this answer - we create an isosceles triangle by drawing a straight line perpendicular to the segment from the origin to the intersection point, and draw straight lines from the origin to its sides. We then look only at one half of the triangle, and receive $\tan(\alpha) = \epsilon \Rightarrow l(\epsilon) = 2 \tan^{-1}(\epsilon)$. The exact ratio, summing the two intersection points, is then $\frac{4 \tan^{-1}(\epsilon)}{2\pi}$ (recall that $r = 1$). From the approximated ratio in Equation (7.4) we receive, $r(2,1,\epsilon) = \frac{2\epsilon}{\pi}$. This can be derived from the exact expression by approximating $\tan^{-1}(x) \approx x$, which is a third-order Taylor approximation. This approximation is illustrated in Figure 7.2. The continuous line represents the identity function $f(x) = x$, where the broken line represents $f(x) = \tan^{-1}(x)$. We can clearly see the two graphs coincide for small values of $x$.

Next, we examine $n = 3$ and $L = 1$. The represented signals lie again on a line that goes through the origin, and intersects the unit three-dimensional sphere in two point (see the middle illustration in Figure 7.1, where only one intersection point is presented). In this case we imagine a cone whose base is the circle with radius $\epsilon$ centered exactly in the intersection point. The head of the cone is in the center of the three-dimensional ball. the surface area on the unit ball that is bounded by this cone equals exactly $2\pi r(r-h)$, where $r$ is the radius of the sphere,
Figure 7.1: Area calculation scheme. On the left - n=2 and L=1, on the middle - n=3 and L=1, and on the right - n=3, L=2.

Figure 7.2: Approximation of the function tan$^{-1}$ using the identity function.
and $h$ is the distance between the center of the sphere and the circle bounded inside both the cone and the sphere (see Figure 7.3). Calculations similar to those done before show that $h = \cos(\tan^{-1}(\epsilon))$. We remind the reader the surface area of a three-dimensional ball is $4\pi r^2$, and therefore the exact ratio, summing up the two intersection points, is $(1 - \cos(\tan^{-1}(\epsilon)))$. The approximation in Equation (7.4) suggests $r(3, 1, \epsilon) = \frac{2}{2}$, which is, again, a third-order Taylor approximation.

![Figure 7.3: Calculation of the surface area of a dome.](image)

Finally, we examine the case when $n = 3$ and $L = 2$. The intersection of a two-dimensional surface with a three-dimensional sphere is a circle of radius 1 whose center is in the origin. We allow a deviation of $\epsilon$ to each dimension, and receive a strip that lies on the surface of this sphere. The exact surface area on the unit sphere, resulted by projecting this strip on the sphere, equals $\sin(\tan^{-1}(\alpha))$. The approximated ratio from Equation (7.4) is $r(3, 2, \epsilon) = \frac{2\pi\cdot2\epsilon}{4\pi} = \epsilon$, which is, again, a third-order approximation of the exact ratio.

In order to finalize this discussion, we calculated the approximated ratio of the represented area in dimension $n = 8$, and subspaces of dimension $L = 3$ with increasing representations errors. We compare our approximated ratio to
a measured one. For this, we chose $L$ atoms in dimension $n$, and generated 10,000,000 normalized points randomly. We then checked for each point whether it is represented by the $L$ atoms with an error no greater than the allowed one. Figure 7.4 shows that up to an error of 16 degrees, the approximated ratio and the true measured one are almost identical.

Figure 7.4: Approximated and calculated ratio, with $n = 8$ and $L = 3$, and increasing error.

In the rest of this discussion we assume the allowed representation error is small enough to neglect the above error. Using this knowledge, a simple theorem can be stated,

**Theorem 1:** In order to cover a ratio $p$ of the whole $n$-dimensional space using no more than $L$ coefficients, with an Euclidean error less or equal $\epsilon$, at least $\frac{p}{r(n,L,\epsilon)}$ subspaces are required.

The proof is trivial, emerging from the above discussion. We will illustrate
a simple usage of this Theorem. Let us consider we are interested in representing at least $p = 0.9$ of all signals in an $n = 3$-dimensional space, using no more than $L = 2$ atoms for each, with an Euclidean error smaller or equal $\epsilon = 0.01$. We calculate,

$$r(n, L, \epsilon) = 0.01,$$

$$\frac{p}{r(n, L, \epsilon)} = 90.$$ 

Therefore a minimum of 90 subspaces is required. Each two atoms can create a subspace, so if we denote the number of required atoms as $K$, we receive the following constraint

$$\binom{K}{2} \geq 90,$$

$$\Rightarrow K \geq 14,$$

That is, the dictionary matrix must be at least of size $3 \times 14$. Any smaller matrix will necessarily derive that the ratio of represented signals is smaller than $p$.

### 7.3 Representation Abilities of a Dictionary

Given a specific dictionary $D \in \mathcal{R}^{n \times K}$, how much of the whole space of dimension $n$ it can represent with an error no more than $\epsilon$ and with no more than $L$ atoms per-representation? We denote the answer to this question by $\rho(D, L, \epsilon)$. Using the results from the previous section, it is easy to see that as long as $D$ includes at least $L$ independent atoms (so that at least one $L$-dimensional subspace can be spanned by it),

$$r(n, L, \epsilon) \leq \rho(D, L, \epsilon) \leq \binom{K}{L} \cdot r(n, L, \epsilon).$$

(7.7)
But those bounds are far from the true value of $\rho(D, L, \epsilon)$. It is obvious that there are overlaps between different subspaces (at least between intersecting subspaces), and therefore, the total covered area is less than $\binom{K}{L} \cdot r(n, L, \epsilon)$. Also, the covered surface area is obviously much more than $r(n, L, \epsilon)$, which represents the area of one single subspace.

In what follows, we develop a tighter lower bound for $\rho(D, L, \epsilon)$, but first, we define some notations. We denote $\sigma_{\min}^s(X)$ (resp. $\sigma_{\max}^s(X)$) as the minimal (resp. maximal) singular value of any set of $s$ columns from the matrix $X$. If $s$ is missing, $\sigma_{\min}(X)$ (resp. $\sigma_{\max}(X)$) will serve as the minimal (resp. maximal) singular values of the matrix $X$. Furthermore, if the matrix name is missing, we refer to the dictionary matrix $D$, $\sigma_{\min}^s = \sigma_{\min}^s(D)$, $\sigma_{\max}^s = \sigma_{\max}^s(D)$. We also define $\sigma_{\max}(X, Y)$ as $\max\{\sigma_{\max}(X), \sigma_{\max}(Y)\}$ and $\sigma_{\min}(X, Y)$ as $\min\{\sigma_{\min}(X), \sigma_{\min}(Y)\}$

**Theorem 2:** Given a dictionary $D$, the ratio of represented signals, denoted by $\rho(D, L, \epsilon)$, allowing $L$ atoms for each representation and a representation error less or equal $\epsilon$ that obeys

$$\epsilon < \frac{\sigma_{\min}}{\sqrt{2}\sigma_{\max}}$$

satisfies the following inequalities:

$$\binom{K}{L} \cdot r(n, L, \epsilon) - \sum_{m=1}^{L-1} \binom{K}{m} \cdot \gamma(m) \leq \rho(D, L, \epsilon) \leq \binom{K}{L} \cdot r(n, L, \epsilon),$$

where we define

$$\gamma(m) = \min\left\{ r(n, L, \epsilon), \frac{S_m(1) \cdot V_{n-m}(R(m))}{S_n(1)} \right\},$$

$$R(m) = \frac{2\epsilon}{(\sigma_{\min}^{L-m})^2 + (\sigma_{\max}^m)^2} \left[ \frac{\sigma_{\min}^{L-m} + \frac{\sigma_{\max}^m}{\sigma_{\min}^{L-m}} \sqrt{(\sigma_{\max}^m)^2 + (\sigma_{\min}^{L-m})^2 - (\sigma_{\min}^{L-m})^2}}{(\sigma_{\min}^{L-m})^2 + (\sigma_{\max}^m)^2} \right].$$
The proof of the above theorem will be built in two steps. First we will define a condition which guarantees that each non-intersecting two sets of \( L \) atoms cover non-overlapping areas on the \( n \)-dimensional hyper-sphere. Then, we will bound from above the overlapping area of two \( L \)-atoms sets whose intersection is not empty. Doing so, we will be able to bound from below the coverage abilities of a dictionary by summing the coverage of all \( L \)-atoms sets, and reducing the bounds for the overlapping areas.

### 7.3.1 Representation by Non-Intersecting Sets of Atoms

**Theorem 3:** let \( D^1 \) and \( D^2 \) be two matrices of size \( n \times L \) with normalized columns, and we denote \( D = [D^1D^2] \).

If there exists an \( n \)-dimensional point \( x \) and two coefficients vectors \( \alpha_1, \alpha_2 \in \mathcal{R}^L \) that obey \( \|D^1\alpha_1\| = 1, \|D^2\alpha_2\| = 1 \) such that \( \|D^1\alpha_1 - x\| \leq \epsilon \) and \( \|D^2\alpha_2 - x\| \leq \epsilon \) then \( \epsilon > \frac{\sigma_{\text{min}}(D)}{\sqrt{2\sigma_{\text{max}}(D^1,D^2)}} \).

In other words, if we set the representation allowed error to be less than \( \frac{\sigma_{\text{min}}(D)}{\sqrt{2\sigma_{\text{max}}(D^1,D^2)}} \), there cannot be any point \( x \) that is represented by both \( D^1 \) and \( D^2 \) with an error less or equal \( \epsilon \).

**Proof:** Let us assume that there exists a point \( x \) and two coefficients vectors \( \alpha_1, -\alpha_2 \in \mathcal{R}^L \) such that \( \|D^1\alpha_1 - x\| \leq \epsilon \) and \( \|D^2(-\alpha_2) - x\| \leq \epsilon \). We then get:

\[
\|D^1\alpha_1 + D^2\alpha_2\| = \left\|D \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \right\| \leq 2\epsilon. \tag{7.11}
\]

We denote the coefficient vector as \( \alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \), and its normalized version,

\[
\tilde{\alpha} = \frac{\alpha}{\|\alpha\|_2}. \tag{7.12}
\]
We divide each side of Equation (7.11) by $\|\alpha\|_2$, and obtain

$$\|D\alpha\|_2^2 \leq \frac{2\epsilon}{\|\alpha\|_2^2}. \tag{7.13}$$

Now, it is true that

$$\|D\alpha\|_2^2 = \hat{\alpha}'D'D\hat{\alpha} \geq \lambda_{\min}(D'D) \Rightarrow \|D\alpha\|_2 \geq \sigma_{\min}(D). \tag{7.14}$$

We conclude that

$$\sigma_{\min}(D) \leq \frac{2\epsilon}{\|\alpha\|_2^2 \left\| \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} \right\|_2}. \tag{7.15}$$

Also, it holds true that for all vectors $x$ and matrices $A$,

$$\|x\|_2 \geq \frac{\|Ax\|_2}{\|A\|_2}, \tag{7.16}$$

where $\|A\|_2$ is the induced norm\(^1\) of the matrix $A$. Using the fact that $D^i\alpha_i$ is also normalized, we get

$$\forall \ i, \ \|\alpha_i\|_2 \geq \frac{1}{\|D^i\|_2} = \frac{1}{\sigma_{\max}(D^i)}, \tag{7.17}$$

and then,

$$\|\alpha\|_2 \geq \frac{\sqrt{2}}{\max\{\|D^1\|_2,\|D^2\|_2\}} = \frac{\sqrt{2}}{\sigma_{\max}(D^1, D^2)}, \tag{7.18}$$

for the above and from (7.15) we get

$$\sigma_{\min}(D) \leq \sqrt{2}\sigma_{\max}(D^1, D^2)\epsilon. \tag{7.19}$$

and this completes the proof. \qed

\(^1\) The induced norm of a matrix $A$ is defined as $\max_x \frac{\|Ax\|_2}{\|x\|_2}$, and for Euclidean vector norms it is proved to be $\sigma_{\max}(A)$. 

The bound in the above Theorem is tight. Let us look at the following example, for which \( n = 5, L = 2 \). We set:

\[
D^1 = \begin{pmatrix}
1 & 1/\sqrt{2} \\
0 & 1/\sqrt{2} \\
0 & -0.001 \\
0.001 & 0
\end{pmatrix} \tag{7.20}
\]

\[
D^2 = \begin{pmatrix}
1 & 1/\sqrt{2} \\
0 & 1/\sqrt{2} \\
0.001 & 0 \\
0 & 0.001
\end{pmatrix} \tag{7.21}
\]

In addition, we set \( \epsilon \) to be 0.0005412, and \( \mathbf{x} = [-0.9239, -0.3827, 0, -0.0005]^T \).

The vector \( \mathbf{x} \) can be represented by the two matrices with an error of epsilon, using the coefficients vectors \( \alpha_1 = \alpha_2 = [-0.5412, -0.5412]^T \). In addition, for this configuration, it turns out that \( \epsilon = \frac{\sigma_{\min}}{\sqrt{2}\sigma_{\max}} \).

Let us now briefly discuss the meaning of the above theorem. Given a dictionary \( \mathbf{D} \), we can guarantee no overlaps between non-intersecting sub-spaces by working with a small enough representation error, which depends on the properties of \( \mathbf{D} \). Setting \( \epsilon \leq \frac{\sigma_{\min}}{\sqrt{2}\sigma_{\max}} \) assures us that no two distinct sub-spaces in \( \mathbf{D} \) overlap.

### 7.3.2 Representation by Intersecting Sets of Atoms

We now explore the possible overlap between two subspaces which share mutual generating atoms. But before we dive into the detailed discussion, let’s first discuss its intuition. Two sets of \( L \)-atoms, with exactly \( m \) mutual atoms must intersect. Think about the surface spanned by \( x \) and \( y \) coordinates and the
one spanned by $y$ and $z$. Each such surface intersects with the unit 3-dimensional ball creating a unit-radius circle. The two circles intersect in two points - $(0, 1, 0)$ and $(0, -1, 0)$. We denote this intersection of the two circles as ‘exact intersection area’, and its shape, in the general case, is of a $m$-dimensional hyper-sphere, where $m$ is the number of mutual atoms. When those two circles become strips, because of the allowed representation error, their intersection widens (see left side of figure 7.5). We denote the new intersection area as ‘widen intersection area’. The latter is the one we are interested in bounding. The bound will be the surface area of the ‘exact intersection area’, multiplied by the volume of an $n - m$-dimensional hyper-sphere whose radius is the maximal distance between a point in the ‘exact intersection area’ and a point in the ‘widen intersection area’ (see left size of Figure 7.5). The only measure which is still unknown is the above maximal distance. This distance actually measures the maximal deviation from the mutual $m$-dimensional sub-space to anyone of the $2L - m$ left possible directions, and therefore reflected by the norm of the coefficients of the non-mutual atoms.

Figure 7.5: On the left, illustration of two overlapping strips. On the right, illustration of the bounding area of the intersection.
Let $D^1D^3 \in \mathcal{R}^{n \times L}$ and $D^2D^3 \in \mathcal{R}^{n \times L}$ be two sets of $L$ atoms from $D$, such that $D^3 \in \mathcal{R}^{n \times m}$, and $D^1, D^2 \in \mathcal{R}^{n \times L-m}$ with distinct atoms from $D$. We further denote $\rho_{2L-m} = \sigma_{\min}( [D^1, D^2, D^3] )$ and $\rho_{L-m} = \sigma_{\min}( [D^1, D^2] )$. An overlap between the two corresponding covered strips includes a point $x$ if,

$$\left\| [D^1, D^3] \begin{pmatrix} \alpha_1 \\ -\gamma_1 \end{pmatrix} - x \right\| \leq \epsilon,$$  \hspace{1cm} \hspace{1cm} (7.22)

and

$$\left\| [D^2, D^3] \begin{pmatrix} \alpha_2 \\ -\gamma_2 \end{pmatrix} - x \right\| \leq \epsilon.$$  \hspace{1cm} \hspace{1cm} (7.23)

As before, we write

$$\left\| [D^1, D^2, D^3] \begin{pmatrix} \alpha_1 \\ -\alpha_2 \\ \gamma_1 - \gamma_2 \end{pmatrix} \right\| \leq 2\epsilon.$$  \hspace{1cm} \hspace{1cm} (7.24)

The left part of the equation is greater or equal to the least singular value of the matrix multiplied by the norm of the coefficient vector,

$$\left\| [D^1, D^2, D^3] \begin{pmatrix} \alpha_1 \\ -\alpha_2 \\ \gamma_1 - \gamma_2 \end{pmatrix} \right\| \geq \rho_{2L-m} \left\| \begin{pmatrix} \alpha_1 \\ -\alpha_2 \\ \gamma_1 - \gamma_2 \end{pmatrix} \right\|,$$  \hspace{1cm} \hspace{1cm} (7.25)

and we can set

$$\left\| \begin{pmatrix} \alpha_1 \\ -\alpha_2 \end{pmatrix} \right\|^2 + \left\| \begin{pmatrix} \gamma_1 - \gamma_2 \end{pmatrix} \right\|^2 \leq \left( \frac{2\epsilon}{\rho_{2L-m}} \right)^2.$$  \hspace{1cm} \hspace{1cm} (7.26)

We now explore again Equation (7.24).

$$\left\| [D^1, D^2] \begin{pmatrix} \alpha_1 \\ -\alpha_2 \end{pmatrix} + D^3 \begin{pmatrix} \gamma_1 - \gamma_2 \end{pmatrix} \right\| \leq 2\epsilon.$$  \hspace{1cm} \hspace{1cm} (7.27)
Applying norm rules and the same argument as before, we obtain
\[
\rho_{L-m} \left\| \begin{pmatrix} \alpha_1 \\ -\alpha_2 \end{pmatrix} \right\| - \left\| D^3 \left( \gamma_1 - \gamma_2 \right) \right\| \leq \left\| [D^1, D^2] \begin{pmatrix} \alpha_1 \\ -\alpha_2 \end{pmatrix} + D^3 \left( \gamma_1 - \gamma_2 \right) \right\|
\]

We formulate the above two equations a bit different, and use the inequality from Equation (7.26), to receive,
\[
\left( \rho_{L-m} \left\| \begin{pmatrix} \alpha_1 \\ -\alpha_2 \end{pmatrix} \right\| - 2\epsilon \right)^2 \leq \| D^3 \|^2 \left[ \left( \frac{2\epsilon}{\rho_{2L-m}} \right)^2 - \left\| \begin{pmatrix} \alpha_1 \\ -\alpha_2 \end{pmatrix} \right\|^2 \right] \tag{7.28}
\]

The above is a quadratic equation, for which the solutions bound the \( \ell_2 \) norm \[ \left\| \begin{pmatrix} \alpha_1 \\ -\alpha_2 \end{pmatrix} \right\|_2 \],
\[
\left\| \begin{pmatrix} \alpha_1 \\ -\alpha_2 \end{pmatrix} \right\|_2 \leq \frac{2\epsilon}{\rho_{L-m}^2 + \| D^3 \|^2} \left[ \rho_{L-m} + \frac{\| D^3 \|}{\rho_{2L-m}} \sqrt{\| D^3 \|^2 + \rho_{L-m}^2 + \rho_{2L-m}^2} \right] \tag{7.29}
\]

The obtained lower bound is negative, and therefore uninformative. However, the upper bound gives us all the information we need in order to bound the overlap area between the two strips.

We received an upper bound on the norm of the coefficient that multiply the non-mutual elements. In order to bound the overlapped area we assume the ‘worst case scenario’ in which the space spanned by the non-mutual atoms is perpendicular to the space spanned by the mutual one, and then the maximal distance from the mutual sub-space is bounded by this norm’s value
\[
\left\| [D^1, D^2] \begin{pmatrix} \alpha_1 \\ -\alpha_2 \end{pmatrix} \right\| \leq \| D^1 \| \| D^2 \| \left\| \begin{pmatrix} \alpha_1 \\ -\alpha_2 \end{pmatrix} \right\| \leq \left\| \begin{pmatrix} \alpha_1 \\ -\alpha_2 \end{pmatrix} \right\| \] (we remind the reader
that the columns are normalized, so the maximal singular value of \([D^1, D^2]\) must be greater than or equal to 1).

The overlapping area will be the surface area of an hyper-sphere of dimension \(m\) with radius 1, multiplied by the volume of an hyper-sphere of dimension \(n - m\) with radius \(\eta(D^1, D^2, D^3)\), where \(\eta(D^1, D^2, D^3)\) is the bound on the coefficient norm, as written in Equation (7.29). This expression is as follows,

\[
S_m(1) \cdot V_{n-m}(\eta(D^1, D^2, D^3)).
\] (7.30)

However, the upper bound can be bounded by itself by the area of one strip, as the overlapped area cannot be greater than any one of the overlapping areas. We therefore receive our final upper bound for the overlapping area of two strips \([D^1, D^3]\) and \([D^2, D^3]\), where the dimension of all atoms is \(n\), the number of columns of \(D^3\) is \(m\), the number of columns in either \([D^1, D^3]\) or \([D^2, D^3]\) is \(L\), and the allowed representation error is \(\epsilon\) by

\[
\min \left\{ r(n, L, \epsilon), S_m(1) \cdot V_{n-m}(\eta(D^1, D^2, D^3)) \right\}.
\] (7.31)

We also define a bound that depends only on the number of mutual atoms in the two strips, by defining \(\rho_{L-m}\) and \(\rho_{2L-m}\) as the minimum singular values as required above, for all possible combination of 2 sets of \(L\) atoms with \(m\) mutual elements.

We can now approximate a more strict lower bound on the coverage of a dictionary \(D \in \mathcal{R}^{n \times K}\),

\[
\rho(D) < \binom{K}{L} r(n, L, \epsilon),
\] (7.32)

\[
\rho(D) \geq \binom{K}{L} r(n, L, \epsilon) - \sum_{k=1}^{L-1} \binom{K}{k} \cdot \gamma(k)
\]
and this completes the proof. \hfill \Box

We demonstrate the bounds we received in graphs. For this, we examine a Grassmanian dictionary $\mathbf{D} \in \mathbb{R}^{n \times K}$, for which we assume the mutual coherence is,

$$\mu = \sqrt{\frac{K - n}{n \cdot (K - 1)}}. \quad (7.33)$$

The mutual coherence $\mu$ is the largest off-diagonal value in the Gram-matrix $\mathbf{D}^T \mathbf{D}$. We use the Gershgorin Theorem [59] in order to bound the maximal and minimal singular values of all sizes matrices,

$$\sigma_{\text{max}}^m \leq \sqrt{1 + m \cdot \mu} \quad (7.34)$$
$$\sigma_{\text{min}}^m \geq \sqrt{1 - m \cdot \mu}$$

Notice those bounds are very pessimistic, because we do not know the exact dictionary, and cannot calculate the singular values exactly. This fact prevents us from examining very redundant dictionaries.

Figure 7.6 displays the upper and lower bounds on the expected ratio of the represented signals under the above terms. As can be seen, those bounds indicate on very limit representation abilities, which we consider as an advantage in Sparseland. Assuming those limited abilities, all we should think of when trying to design a dictionary is an adequate representation of the signals of interest, while the inability to represent other signals is almost guaranteed.

### 7.4 Summary

In this section we studied the relative volume of signals having a sparse representation and a permissible error. However, much more remains to be done
Figure 7.6: Illustration of the coverage of a Grassmanian dictionary of dimension 10 for several redundancy levels
in order to use these theorems in the design of dictionaries. The main objective should be to maximize the intersection between those represented signals and the set of signals we desire to represent sparsely. Thus, localizing the covered area on our geometrical sphere in the desired place.

Such a process is a great challenge that can eventually lead to an optimal dictionary - a dictionary that can sparsely represent the signals of interest, and in the same time, cannot represent sparsely other signals.
Chapter 8

Summary and Conclusions

8.1 General

This thesis concentrates on the subject of dictionary learning for sparse representation of signals, based on the Sparseland model we have introduced. Under this model, signal families (such as images) are assumed to be well represented as sparse linear combinations of atoms from a predetermined dictionary. Naturally, we expect that designing the appropriate dictionary will result with better treatment of such signals, and lead eventually to better handling of them in various problems, such as their reconstruction in inverse problems, their compression, and more.

For this purpose, a novel algorithm – the K-SVD – is introduced in this work. This algorithm can efficiently train a dictionary from a given set of signals it aims to represent. The K-SVD is the main contribution of this work, and as such, it stands in its center. Our main aim in this work was to show that beyond its ability to outperform its alternatives, such as the MOD algorithm, it can be incorporated to practical applications in image processing and lead to unprecedented performance. State-of-the-art denoising results are achieved using the K-SVD, when combined to a MAP-based approach that turns local
representation of patches into a global prior on the whole image.

Variations of this algorithm for structural constrained dictionaries are also described in our work. Such variations are crucial for handling special cases of interest, where the *Sparseland* model goes through some modifications. Such is the case, for example, with the non-negative factorization option, and also with a multiscale representation. Unfortunately, beyond putting the foundations for such extensions, our work does not lead to successful applications using these modified K-SVD methods.

We believe the field of sparse representation by data-driven dictionaries has a great potential, and the excellent results presented so far for denoising are only the first among many to come.

### 8.2 Future Work

Many future directions for the above-described work exist. We shall list here several such directions that we find as promising and intriguing:

- Building a firm model for representation of full size images, and proving its applicability, is a great challenge. In this context, there is still much room for improvement when it comes to a multiscale version of the K-SVD, as we have outlined in this work. Indeed, a conceptual multiscale representation using dictionaries of small sizes as considered here and pursuit techniques to accompany it, both stand as major challenges. A proper answer to these issues is likely to lead to much better handling of signals.

- Applying trained dictionaries for other applications such as compression, inpainting, classification, pattern detection, denoising for video and color
images, and more, are all interesting directions that could and should be explored.

- We proved uniqueness of the dictionary in the case of an exact representation. We expect that when approximated representation is assumed, uniqueness does no longer exists. However, our experiments indicate that a stability of the underlying dictionary does hold. Proving such a theorem is not trivial, and is left for future work.

- When we discuss the coverage abilities of a given dictionary under known representation and sparsity constraints, we did not refer to the most important question of localizing this covered area in the desired place. That is, maximizing the intersection between the set of represented signals by the designed dictionary and the set of signals we desire to represent sparsely. Such a study is a great challenge that can eventually lead to an optimal dictionary - a dictionary that can sparsely represent the signals of interest, and in the same time, cannot represent sparsely other signals. Such analysis could also give a measure of quality to the model, and enable its comparison to competitive methods, such as Markov random Field (MRF) models.

- The Sparseland model is arbitrary and it is definitely not perfect. How can it be extended to better describe signals? How such modifications can be incorporated to the K-SVD algorithm? These questions are yet to be explored, and have great potential in providing future techniques in signal processing.
Bibliography


אנלוגים ל-K-SVD - ממציאות גיאית של ריצוף 알 סכום קהל وخופה לחית. הדילוק בין המטרוד יס עליון של שלילה על אביה שלילון, בעיה שיתוף לקשר nào אחר במענה מנובחה.

(Non-negative matrix factorization) או השילוב שלús יס עליון של שלילה על אביה שלילון, בה כל יחס מה IData שילוב יס עליון שלילון, ש實משי

לחסוך או דוגמ נэтажת שלילון של הנמצאות של יס וסילון של יס עליון שלילון, ותרשים המובן וההמירה של הקבוצה.

האיתות הרגולאציה ליצוג תומוק, וה:invokeים סרטים התחומים את יס של קבוצת ותרשים במובן קבלוק.

לימים, עבורה ואני מתכונ עליון של יס וסילון של יס עליון שלילון, עד את יס עליון של יס עליון שלילון, ותרשים במובן קבלוק.

המשתנה או מתכונת של נחוג קהל ותרשים קבלוק.
תקציר

ירזך את מחקרו התזה מילווס ייחודי, המכיל מ neurop defenders צודני גלול המחלה, ומפואר של שיטה בקודقضي

של העקרונות הדומים לשנים ברוח מקוצרת מודל שיבוץ פיזיוסטרו, במכס נפוץ במדינת ישראל. ירזך את מחקרו התזה מילווס ייחודי, המכיל מ neurop defenders צודני גלול המחלה, ומפואר של שיטה בקודقضي

מידה של מודל בלעדי '_sparseland Model' המבוסס על מודל משמשת מודרנט, למורת עיגה לגזרי במודל. המדיה המודל ניתן באמצעות הכלים האפשריים של התפה הדילל מפורחת. ראש החיבור של כל מתמטיקה היברדי בין אגן (למשל, סכון שיווקספיאה),쳇 מודל הדילל על הפיזיוסטרו התזה מילווס ייחודי, המכיל מ neurop defenders צודני גלול המחלה, ומפואר של שיטה בקודرضى

בשובות שמודל התפה, נית הגולו בשורת עיגה מרובית – מציאת הדיללrial ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצטמן מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודי, מודל במודל המודל, באצזמנ מודל ייחודית המודל המשמש לקירחון, המש痂 את הלשון המודל. ושארם ממילון שלימיות סטוקש עוביות מחוון תורכי ניטול שלימיות לבלוג, שאית חסונ המודל משמש לקירחון. ומילון למילון שלימיות סטוקש עוביות מחוון תורכי ניטול שלימיות לבלוג, שאית חסונ המודל משמש לקירחון. ומילון למילון שלימיות סטוקש עוביות מחוון תורכי ניטול שלימיות לבלוג, שאית חסונ המודל משמש לקירחון. ומילון למילון שלימיות סטוקש עוביות מחוון תורכי ניטול שלימיות L-

7.1 מכמות והשורש החסרים של \( \tan^{-1} \). השדרה \( L=2, n=1, \alpha = 3 \).
7.2 הרכזת הקרובים הפונקציות \( \tan^{-1} \). השדרה \( L=1, n=1, \alpha = 3 \).
7.3 משך של הפעמים \( \tan^{-1} \). השדרה \( L=3, n=1, \alpha = 3 \).
7.4 גלגל הנגזרת של \( \tan^{-1} \). השדרה \( L=2, n=1, \alpha = 3 \).
7.5 משך \( \tan^{-1} \) של \( \tan^{-1} \) של \( \tan^{-1} \). השדרה \( L=2, n=1, \alpha = 3 \).
7.6 בנימין (Grassmanian Frame) הויתון \( \tan^{-1} \) של \( \tan^{-1} \) \( \tan^{-1} \). השדרה \( L=2, n=1, \alpha = 3 \).
A = dx^T

NN-K-SVD

K-SVD

SI-K-SVD - ISL-DLA, SVD

(30 dB) MS-K-SVD

רשימה הזוכי

40 K-Means
41 אלגוריתם
42 K-SVD
43 אלגוריתם
44 האתחנה הפרימטרים: עיבור כל אחד מפרימטרות המבוקש עד שייה
45 ברзван 50 ריצות שטורפה-זית מגוון. המפרים הממוצעים של המסרф המוסבר
46 לש האותミニ בתוכן (מחזור מספר 50 אוטומטי קיוםס) עיבור כל 10 ההרצות
47 קצב שתיית 50 בלקים אקראית מתרק קבוצת האימונים, מומרים "ע""וניה"" הנשיגה.
48 המריצי האימונים. האוטומטיםuzzi ע""ץ"" עץ השוגה במעל 50, המפרים כל שערפ.
49 האמור בפרסים להמרות המקסימלי
50 מילון Haar
51 מילון DCT
52 מילון yarn
53 התאמת שיווק השראות המורכבות הממוצעות średni 594 בלקים עם פיקסלים
54 (RMSE) התמונה המרופדת, מילון Haar\
55 התמונה המרופדת, מילון DCT\
56 התמונה המרופדת, מילון Haar\ ייחודי מילון חיתו
57 התמונה המרופדת, מילון DCT\ ייחודי מילון חיתו, השראה החיפその他 וה обучות
58 התמונה המרופדת, שראות בחיפその他 וה обучות, עם עבורי 50% עד 70% של פיקסליםIFF
59 התמונה המרופדת, ייחודי מילון חיתו, השראה החיפその他 וה обучות, עם עבורי 50% עד 70% של פיקסליםIFF
60 הקחז
61 מוקד מתאצות החודש
62
63 שולחן הקורא העיש ע""פי שמו של בלקם המרחוק על התמונה המופיעה בESC:
64 C=1.15 OMP \( \frac{\lambda}{\sigma} = 30, J = 10 \)
65 מילון DCT
66 ייחודי
67 מילון גלובל
68 רוט첬 מתאצות השראות של פיקסלים של בלקם
69 התמונה המרופדת, מילון DCT\ ייחודי מילון חיתו, השראה החיפその他 וה обучות
70 התמונה המרופדת, מילון DCT\ ייחודי מילון חיתו, השראה החיפividad וה обучות, עם עבורי 3 התמונה
71 מוסבר
72 השיפה באלגוריתם בדודדód המרות לאחר כל אוטומטי של אלגוריתים- K-SVD
73 על בלקם המורכבות הממוצעות 'Peppers'
74 'Peppers'"" השופע
75 מתאצות החודש השראות של פיקסלים של בלקם 'Barbara'
76 מתאצות החודש השראות של פיקסלים של בלקם 'Barbara'
77 שופעים, ושתי גרסאות משותפות.

רשימת טבלאות
5.1 סכום תזאת ה-PSNR בحرية הფיתוח: 
DCT וורירית בין-סימטרית: תזאת של [87] Portilla et al. [87] Portilla et al.]

5 התוויות: מילון

יתוי: שיאליות תשים: ממוצע גלואלילי. ממוצע התוויות: תזאת המילון המיאומגעל על התוויות המיאומגעל. לכל ביטויי התזאת וניתוח בתיור מודגשת. שלושה ושישה התוויות המיאומגעלות 

6.1 תזאת המיאומגעל של תוויות אוטומטים: גיבור גייס תכניות של אוטומטים מילונים ומרוניםカフェאrotch. 

מספר האוטומטים הערז 9.1447ibir מבנה מילון בפרט (המקור בנויה), 1-2 הביאו. 

כל וייט בצד 5 פנים.
K-SVD .6

(NN-K-SVD) இ K-SVD 6.2

100

(NT-K-SVD) இ K-SVD 6.3

100

S-pH

129

(Sparsland) .7

139

8.1

150

כרובים
топק עניינים

1. יוצובג דילל של אורות והופות
   1.1 מהקמה וסימון
   1.2 מפרקית נהיה
   1.3 מagedList שילולדה
   1.4 הצורה של האורות
   1.5 מסכת העיתוד

2. סובד וקדרות
   2.1 ייצוגים שונים
   Orthogonal Matching Pursuit (OMP) 2.1.1
   Basis Pursuit 2.1.2
   FOCUSS 2.1.3
   2.2 ניסוח הפוארטו של אלגוריתמים
   2.2.1 חזרות ניסוח
   2.2.2 העיבוד הנווטה
   2.3 מבנים שונים
   2.4 K-Means - מתלטלות של אלגוריתמים
   2.5 שיטה הפרידום המإجراءات
   2.6 סיסםumped החומרים

3. יהודית ענילך
   3.1 ייצוגים שונים
   3.2 ניסוח הגרסת היישודת
   3.3 מבנים שונים
   3.4 סיסםumped

4. אלגוריתם מ-
   K-Means - מ
   K-SVD - מ
   K-Means - מ
   K-SVD - מ
   K-SVD - מ
   4.1 עניב ערכים
   4.2 ליניאריות
   4.3 תיאור כ飽
   4.4 פיזור מוסרי
   4.5 תיאור מוסרי
   4.6 תיאור מוסרי
   4.7 תיאור מוסרי

5. שמוש מ-
   K-SVD - מ
   5.1 תיאור אזור רעש במעון

המחקר שעשה בכותב
ד"ר מיכאל אלעד
בסקוללה המודעי המוסבר

برزוני להודות למונהיה של י"ר מיכאל אלעד, על האמנת
התרומת שכתה לי לכל אורח התמורה. גישה היסודית
המעניקה, לצדו התובה והटילתה בנקת לכל נושא המחקר
הכין את ליימויה הווה.

توجه מיודות לפורפ', אלפרד ברוקשתין, שלחוות אוחי לכל
اورר ליימויה והיה שותף פעל במחבר זה.
אני מודה לcelandות על התמיכה הספירה והנידיה במשלמותי
לبوتון, אני מודה לبطل', חונך. על שחריה מתמיד ליידי והיוואר
הזה מוקדש לו באיהה.
מילים חיות ליצוג דליל
של אוחות

דבורה של פוקר
乐视 מילויים
של הדירוג
לכבות
הנאות דוקטור
לפילוסופיה

מיכל אתרון

הוגש לסמינר הוכני
- מכון תכנולגי
לישראל
נובמבר 2006
חיפה
כשל תשבץ"ז

מילונים תר込み לייצוג דיל
של אזהות

מיכל אוחרן