# Dictionary Learning for Audio Inpainting

Corentin Guichaoua

# Dictionary Learning for Audio Inpainting

Internship report

Corentin Guichaoua

Master informatique – spécialité Recherche en informatique
IRISA/INRIA Tutors (METISS Research group) : Rémi Gribonval and Nancy Bertin


# Contents

# Notations

| Vectors | Dim. | |
|---|---|---|
| $\underline{w}$ | $V$ | placeholder vector for this table |
| $\underline{w}_i$ | 1 | $i$th element of vector $\underline{w}$ |
| $\underline{\gamma}$ | $M$ | sparse coefficients for a single frame |
| $\underline{g}$ | $L$ | sparse coefficients for a single atom |
| $\underline{d}$ | $N$ | atom |
| $\underline{y}$ | $N$ | original signal |
| $\hat{\underline{y}}$ | $N$ | estimation of the original signal |
| $\underline{x}$ | $N$ | single input signal |
| $\underline{r}$ | $N$ | residual term (noise or model mismatch) |
| $w_a, w_s$ | $N$ | analysis window (discrete function) |
| $w_s$ | $N$ | synthesis window (discrete function) |
| **Matrices** | | |
| $\boldsymbol{W}$ | $V \times W$ | placeholder matrix for this table |
| $\boldsymbol{W}_I$ | $V \times \lvert I\rvert$ | sub-matrix of $\boldsymbol{W}$ with the columns indexed by $I$ |
| $\boldsymbol{W}_{I,J}$ | $\lvert I\rvert \times \lvert J\rvert$ | sub-matrix of $\boldsymbol{W}$ with the columns indexed by $J$ and lines indexed by $I$ |
| $\boldsymbol{W}^T$ | $W \times V$ | $\boldsymbol{W}$ transposed |
| $\boldsymbol{D}$ | $N \times M$ | dictionary |
| $\boldsymbol{D_0}$ | $N \times M$ | initial dictionary for dictionary learning algorithms |
| $\boldsymbol{G}$ | $M \times M$ | Gram matrix of $\boldsymbol{D}$ ($\boldsymbol{G} = \boldsymbol{D}^T\boldsymbol{D}$) |
| $\boldsymbol{I}$ | $V \times V$ | identity matrix (size is deducted from context) |
| $\boldsymbol{X}$ | $N \times L$ | matrix of input signals, arranged by columns |
| $\boldsymbol{R}$ | $N \times L$ | reliability mask: $\boldsymbol{R}_{j,i} = 1$ if sample $\boldsymbol{X}_{j,i}$ is clean, 0 otherwise |
| $\boldsymbol{\Gamma}$ | $M \times L$ | sparse coefficients for all frames |
| $\boldsymbol{M}^m, \boldsymbol{M}^r$ | $N \times L$ | Decimation matrix: discards reliable (resp. missing) samples (see below) |
| $\boldsymbol{M}_f^m, \boldsymbol{M}_f^r$ | $N \times L$ | Square decimation matrix: sets reliable (resp. missing) samples to 0 (see below) |
| **Scalars** | | |
| $Z$ | 1 | number of iterations to run |
| $K$ | 1 | maximum number of atoms to use in a sparse representation |
| $N$ | 1 | window size (in samples) |
| $M$ | 1 | number of atoms in the dictionary |
| $L$ | 1 | number of training signals |
| **Misc.** | | |
| $\underset{i \in I}{\operatorname{argmin}} f(i)$ | n/a | an element of $I$ that minimizes $f$ |
| $\boldsymbol{W}^r, \underline{w}^r$ | n/a | $\boldsymbol{W}(\underline{w})$ restricted to reliable samples |
| $\boldsymbol{W}^m, \underline{w}^m$ | n/a | $\boldsymbol{W}(\underline{w})$ restricted to missing samples |
| $\circ$ | n/a | element-wise matrix multiplication |

# 1 Introduction

Restoration of distorted data has been a major concern in signal processing ever since audio was digitized. There are many ways a signal can be distorted, from the moment it is recorded until the moment it is played. This distortion can occur on purpose when trying to find new sounds (in which case it is fine), as a side effect of some processing or because of malfunctions.

During this internship, I focused on distortions that occur locally, where the process of restoration consists in filling the gaps between the parts that are known to be correct. More specifically, based on work that has been completed within the METISS Project-Team, I have attempted to use machine learning to perform audio reconstruction more accurately.

In a first time, I will briefly introduce the institute in which I performed my internship and the team that has welcomed me for its duration. In section 3, I will present in more detail the context and applications of my work. Sections 4 and 5 focus on the presentation of existing work, with further detail on the models that I used. A new algorithm is also introduced in section 5.2. In section 6, I will describe the experimental system and the experiments I conducted with it. Finally, section 7 will address the research axes that have been planned, but have not yet had or will not have the time to be explored before the end of this internship.

# 2 Research center presentation

## 2.1 Inria Rennes

Inria[1] is a public research institute which focuses on computer science and applied mathematics. As of December 2010, it totaled 3,429 scientists among 4,290 people across all 8 centers. The *Rennes – Bretagne Atlantique* center shares buildings, research groups and support departments with the Irisa[2], a joint research unit between the CNRS, the University of Rennes 1, the Insa of Rennes and the Brittany antenna of ENS Cachan.

## 2.2 METISS Project-Team

METISS is one of the joint Inria-Irisa project-teams. Its main research topics are source separation, speech processing, sparse modeling techniques and music information retrieval. The project-team is involved in several contracts with industrial partners and many international projects. As of this writing, it is composed of 4 permanent researchers, 5 engineers, 4 postdoctoral researchers, 4 PhD students, 1 administrative assistant and 2 interns (including myself); during my stay, 1 engineer and 1 postdoc left, 1 engineer and 1 intern arrived. Among the postdocs and PhD students, 4 are not francophone (Canadian, German, Polish and South Korean), which makes English the working language for several projects, as well as an important language for coffee breaks.

[1] `www.inria.fr`

[2] Institute for Research in Computer Science and Random Systems, `www.irisa.fr`

## 3 Context and application

Automatic inpainting is a task that has been studied in image processing [Bertalmio et al., 2000], consisting in filling the gaps of an incomplete image. It has applications for the removal of unwanted objects from the image (such as incrusted text–fig. 1– or the cage in fig. 2) or the restoration of damaged pictures (stains, burns, overexposure, *etc.*).

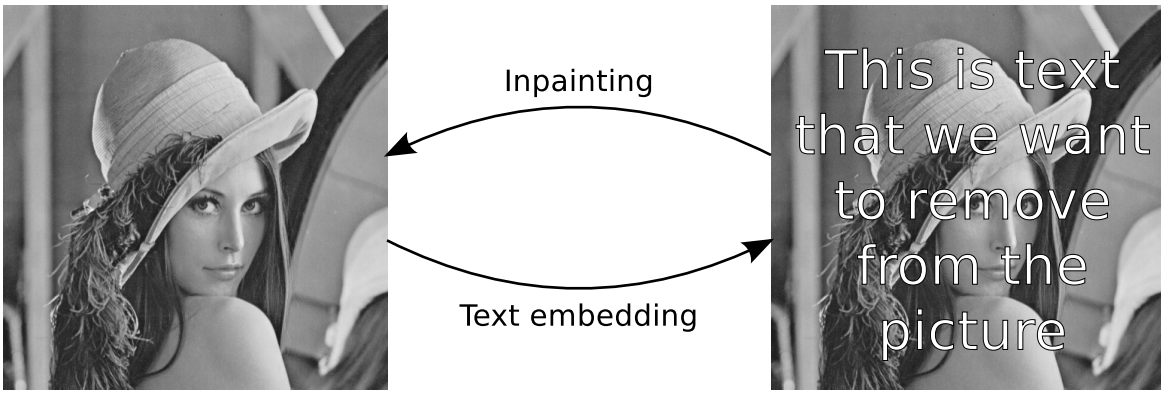


Figure 1: Ideal inpainting on text removal

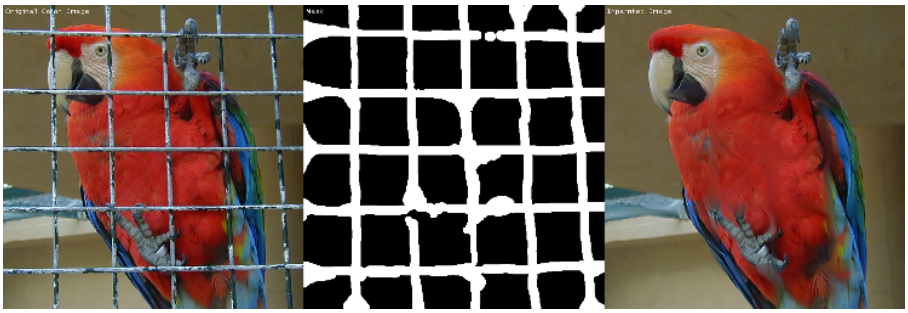

Figure 2: Removal of an undesired object (the cage). From left to right: input image, mask of pixels to estimate, restored image

This concept has been extended to audio in [Adler et al., 2011] in order to regroup a number of existing tasks under a single formulation. Corruption in audio signals has many possible sources:

- network transmission errors may lead to missing packets (5 to 60 ms);
- flaws on the physical support of data (vinyls, scratched CDs) or power surges during acquisition often cause impulsive noise (perceived as clicks);
- source separation algorithms often generate tracks with missing time-frequency regions;
- lossy compression algorithms sometimes discard high-frequency content in high compression modes;

- and careless processing or excess of the nominal range of acquisition devices can cause clipping, *i.e.* truncation of the waveform at a maximum absolute value (see fig. 3).

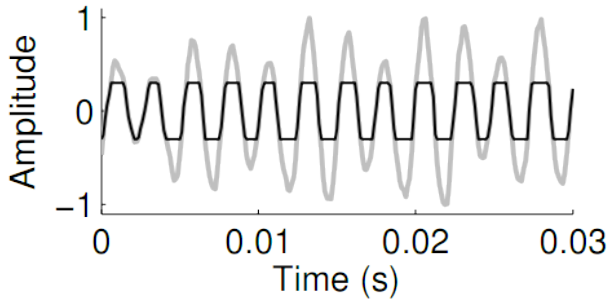


Figure 3: An example of clipping (black: observed signal, grey: clean signal)

As a consequence of this diversity, many very different approaches have been established, using different terminology depending on the application. Most problems referred to as audio interpolation, audio extrapolation, missing sample imputation, bandwidth extension or even packet-loss concealment (for audio streams) can be expressed in a single formulation, presented in the next section.

# 4 Approaches to audio inpainting

## 4.1 Overview

The reversal of a damaging process can be treated as inpainting if:

- the observed data is locally identical to the source (this locality can be either in the time of time-frequency domain),
- the partition between distorted and clean data is known.

Let $\underline{y} \in \mathbb{R}^N$ be the undistorted signal that we want to recover. Based on the two above statements, we can state

$$\boldsymbol{M}^r \underline{x} = \underline{x}^r = \boldsymbol{M}^r \underline{y} \tag{1}$$

where $\underline{x}$ is the observed data. $\boldsymbol{M}^r$ and $\boldsymbol{M}^m$ are the decimation matrices for respectively the reliable and the distorted parts of the data, built by removing the corresponding lines of the identity matrix (hence reducing dimension). $\boldsymbol{M}_f^r$ and $\boldsymbol{M}_f^m$ are built by setting the corresponding values of the identity matrix to 0 and do not reduce dimension. For example, in dimension $N = 4$ with samples 2 and 4 distorted,

$$\boldsymbol{M}_f^r = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } \boldsymbol{M}^m = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

This problem is underdetermined, as there is an infinity of possible estimates $\hat{\underline{y}}$ that will verify the equation: $\underline{y}$ and $\underline{x}$ are two of them but other solutions $\hat{\underline{y}}$ can still be very different. To choose a good solution, we need prior information about $\underline{y}$; this information comes in the form of a model of the signal, which constrains what values it is able to take.

Models that have been used to solve the inpainting problem differ depending on the application, and therefore on the domain and distribution of the distortion:

- Packet-loss is easy to detect during a transfer and is often addressed by re-sending the packet, intervening at a higher level of treatment. However for real-time applications (phone, VoIP), low latency requirement forbids the use of this solution. Furthermore, the length of the missing parts – 5 to 60 ms, the highest of inpainting applications –, makes accurate reconstruction hard. Most algorithms to solve this kind of deterioration are therefore relatively simple, aiming at concealment of the loss rather than its reconstruction[Ofir et al., 2007].
- Impulsive noise removal is an easier problem since the missing blocks are very short and has been addressed in many ways, for example with autoregressive modeling [Janssen et al., 1986, Etter, 1996] or Bayesian estimation of the missing samples [Godsill and Rayner, 1998].
- Declipping presents its own challenges: the distribution of missing samples is not random since only the highest energy samples are missing. On one hand this means that the missing samples distribution conveys information about the original signal, on the other hand the missing samples are the ones that are likely most informative. The problem has been addressed with Bayesian estimation [Godsill et al., 2001], use of band limits [Abel and Smith, 1991] or linear prediction [Dahimene et al., 2008].

In [Adler et al., 2011], all of these problems are addressed with a sparse model, which is described below.

## 4.2 Sparse modeling

### 4.2.1 Model

A sparse model works on the assumption that a signal (or part of it) $\underline{y}$ can be represented as the sum of few elementary pieces of signal, called *atoms*. Mathematically speaking, this translates to

$$\underline{y} = \boldsymbol{D}\underline{\gamma} \tag{2}$$

with $\underline{\gamma} \in \mathbb{R}^M$ sparse, *i.e.* it has few non-zero coefficients. $\boldsymbol{D} \in \mathbb{R}^{N \times M}$ is called the *dictionary* of atoms. For noisy measurements, we have to account for the noise by adding an error term $\underline{r}$:

$$\underline{y} = \boldsymbol{D}\underline{\gamma} + \underline{r} \tag{3}$$

This model is known to apply to a lot of real world signals. For example, most real-world images are approximately sparse on a wavelet basis (used by most modern image codecs), and the sound of a vibrating string is sparse on a Fourier basis since Fourier atoms are the elementary solutions of Helmholtz's wave equation.

When we apply the sparse approximation model to that of inpainting (eq. 1), we obtain

$$\underline{x}^r = \boldsymbol{D}^r\underline{\gamma} + \underline{r}^r \tag{4}$$

where $\boldsymbol{D}^r$ is the dictionary restricted to the observed samples. We can then estimate the sparse coefficients $\hat{\underline{\gamma}}$ on the observed samples and rebuild an estimate of the original signal $\hat{\underline{y}}$ with the full dictionary:

$$\hat{\underline{y}} = \boldsymbol{D}\hat{\underline{\gamma}}. \tag{5}$$

### 4.2.2 Dictionary

The dictionary used in eq. (2) is crucial to the success of the model. A bad choice of dictionary will result in a bad modeling of the signal (either many atoms or a large error term). Conversely, choosing a good dictionary will allow for sparser representations of the signal.

Using a complete dictionary (*i.e.* a basis, $M = N$, all atoms linearly independent) guarantees that the representation is unique, however this does not mean that the signal will be sparse. Popular complete dictionaries are the ones obtained by mathematical transforms, the most well-known of which are the Fourier transform, the short-term Fourier transform (or Gabor transform), wavelet transforms and discrete cosine transforms (DCT).

How good a dictionary is depends greatly on the signal to analyze: a Fourier transform dictionary will have excellent sparsity with a sum of pure tones, but terrible performance with a set of clicks, whereas a waveform representation will obtain opposite results. Therefore, learning the dictionary on a set of similar signals is an interesting option, detailed in section 5.

The uniqueness of the representation with complete dictionaries is an interesting property for the selection of the representation, however it also means that the expressiveness of the dictionary is limited. The use of overcomplete dictionaries ($M > N$) can allow for sparser representations by introducing redundancy: the extra atoms can be seen as shortcuts for a linear combination of other atoms. Unfortunately, with this increase in expressiveness comes the need to select the best possible representation, as equation (2) becomes underdetermined.

A common way to obtain overcomplete dictionaries is to use a transform basis with a higher sampling rate, or to use several smaller bases at overlapping locations in the signal.

In this report, the dictionaries we will use will have atoms with a unit norm. This allows for simplifications in the algorithms and can be enforced through a normalization of the dictionary without changing the way it models signals.

#### 4.2.3 Sparse coding

There are two essential properties of a sparse approximation: its sparsity and its fidelity. The sparsity is how few of the coefficients used in the representation are different from 0, which is measured by the $l^0$ pseudo-norm $\|.\|_0$, while the fidelity represents how close the approximation is from the actual signal, and is typically measured with the Euclidian norm. Since those two criteria usually vary in opposite directions (increasing sparsity means decreasing fidelity and reversely), we need to find a trade-off. Using a cost function $\mathcal{L}$ that formalizes this trade-off, the problem to solve is:

$$\underset{\underline{\gamma}}{\operatorname{argmin}} \mathcal{L}\left(\underline{\gamma}, \underline{x}^r - \boldsymbol{D}^r.\underline{\gamma}\right). \tag{6}$$

Optimally, $\mathcal{L}$ measures the sparsity with the $l^0$ norm, however since the $l^0$ norm is non-convex, the problem is NP-hard in general [Davis et al., 1997]. To achieve tractable results, we have to settle for a sub-optimal solution. Two branches of methods are available in order to find such a solution: relaxing the $l^0$ norm or using a greedy algorithm.

**Relaxed solutions** By relaxing the $l^0$ pseudo-norm into a convex norm, such as its closest convex approximation the $l^1$ norm (a.k.a. Manhattan distance), convex optimization algorithms can be used, and the result can be mapped to a sub-optimal sparse representation. These methods solve one of the following problems:

$$\underset{\underline{\gamma}}{\operatorname{argmin}} \left\|\underline{x} - \boldsymbol{D}.\underline{\gamma}\right\|_2 + \lambda \left\|\underline{\gamma}\right\|_1 \tag{7}$$

$$\underset{\underline{\gamma}}{\operatorname{argmin}} \left\|\underline{\gamma}\right\|_1 \text{ s.t. } \left\|\underline{x} - \boldsymbol{D}.\underline{\gamma}\right\|_2 \leq \epsilon \tag{8}$$

$$\underset{\underline{\gamma}}{\operatorname{argmin}} \left\|\underline{x} - \boldsymbol{D}.\underline{\gamma}\right\|_2 \text{ s.t. } \left\|\underline{\gamma}\right\|_1 \leq K. \tag{9}$$

Examples of this kind of algorithm are the Majorization-Minimization method [Lange et al., 2000] or FISTA [Beck and Teboulle, 2009].

**Greedy pursuit** An approximate solution can also be found using a greedy algorithm, selecting atoms one after another. Some of the available ones are:

**Matching Pursuit (MP)** Proposed in [Mallat and Zhang, 1993], this algorithm is the base of most pursuit algorithms. As described in algorithm 1, at every step, it selects the atom that is most correlated to the residual, then removes the projection of the residue onto this atom in order to get the next residual.

**Orthogonal Matching Pursuit (OMP)** Proposed in [Pati et al., 1993], OMP changes the residual update step of MP by projecting the residual on *all* the selected atoms (instead of only the last one). This projection is much slower, but the algorithm converges in fewer iterations and therefore leads to a better solution (smaller error term for the same number of atoms).

**BatchOMP** This algorithm is a slight variant of OMP, proposed in [Rubinstein et al., 2008a]. It takes the Gram matrix of the dictionary[3] as an extra argument and uses it in order to run OMP quicker. The cost of computing the Gram matrix is quickly offset if the sparse decomposition is performed on multiple signals with the same dictionary.

**LocOMP** LocOMP is another implementation variant of OMP, presented in [Mailhé et al., 2009], which focuses on dictionaries with local atoms, *i.e.* atoms that are much smaller than the signal to represent. Under this assumption, it can decompose a signal in the order of magnitude of real-time, scaling log-linearly. This makes it possible to work on a full-length signal (as opposed to a framed signal) or with a shift-invariant dictionary.

Other greedy algorithms exist, such as Gradient Pursuit [Blumensath and Davies, 2008] or Stagewise-OMP [Donoho et al., 2006].

**Algorithm 1** The Matching Pursuit algorithm [Mallat and Zhang, 1993]

**input:** $\underline{x}$; $\boldsymbol{D}$
**output:** $\underline{\gamma}$
$\underline{r} = \underline{x}$
$\underline{\gamma} = \underline{0}$
**repeat**
  $k_{opt} = \underset{k \in \{1..M\}}{\operatorname{argmax}} |\langle \underline{r}, \boldsymbol{D}_k \rangle|$
  $c = \langle \underline{r}, \boldsymbol{D}_{k_{opt}} \rangle$
  $\underline{r} = \underline{r} - c.\boldsymbol{D}_{k_{opt}}$
  $\underline{\gamma}_{k_{opt}} = \underline{\gamma}_{k_{opt}} + c$
**until** $\underline{r}$ small enough or number of iterations reached

Both approaches are computationally costly and sub-optimal under general conditions. Compared to convex norm minimization approaches, greedy algorithm tend to fall in local minima, *i.e.* locally optimal solutions that algorithms can not escape, however, they are faster and easier to adapt to incorporate extra information on the signal (as for declipping)[Mallat, 2009, Ch.12]. Thus, we chose to use this branch of algorithms. Specifically, we used BatchOMP because learning dictionaries involves the conditions it has been designed for.

[3]The Gram matrix of a matrix $\boldsymbol{D}$ is $\boldsymbol{G} = \boldsymbol{D}^T\boldsymbol{D}$, *i.e.* the matrix of correlations between columns (the atoms in the case of a dictionary)

#### 4.2.4 Declipping adaptations

For declipping, additional information provided by the location of samples: corrupted samples have absolute values that exceed the clipping level, and their sign can be determined from the corrupted value. In order to incorporate this knowledge in the inpainting process, an additional step is added at the end of the sparse approximation: the non-zero coefficients are recomputed with constraints so that negative clipped samples are reconstructed below the negative clipping value, and positive clipped samples are reconstructed above the positive clipped value. Additional constraints are also added so that the samples do not take excessively high values. This maximum value can be computed by an estimation of the clipping level.

### 4.3 Our focus

Audio declipping is the case of audio inpainting consisting in the reconstruction of saturated samples. In recent work, this problem has been approached as solving a linear inverse problem with additional constraints, using sparse modeling of the clean parts of the signal. One of the key points of sparse modeling is the dictionary used for the representation. We study the impact of the choice of the dictionary for this method of inpainting, focusing on the comparison between chosen dictionaries and learnt dictionaries. We also propose a variation of the K-SVD dictionary learning algorithm, adapted to learning on locally corrupted signals.

# 5 Dictionary learning for audio declipping

Dictionary learning consists in building a dictionary, based on a set of training signals, so that the dictionary will allow for sparser representations when it is used on signals that are similar to the ones it has been trained on. The choice of the signals to train on is a crucial step: if it is too homogeneous, the dictionary will be very specialized and unable to represent slightly different signals and if it is too diverse, the dictionary will not be specific enough to gain much through the learning process.

A workaround to this choice is the approach of *bootstrapping*. This consists in learning when the application task is presented to the algorithm, from the application examples. However, since the application examples are incomplete data, this requires special algorithms. An example of this kind of approach can be found in [Zeyde et al., 2010] for image scale-up (estimating a higher resolution version of an image): the input image is scaled down further in order to generate a training set and the learned dictionary is used to build an estimate of the high resolution image. We present an algorithm able to bootstrap on audio inpainting tasks in subsection 5.2.

## 5.1 Existing dictionary learning algorithms

### 5.1.1 Overview

Several algorithm exist to learn a dictionary for sparse approximation. K-SVD [Aharon et al., 2006], detailed in the next subsection, and MOD [Engan et al., 1999] are based on iterative processes similar to those used in codebook learning for vector quantization (K-Means and such). Some other approaches attempt to train dictionaries with special properties, like translation invariance [Mailhé et al., 2008, Blumensath and Davies, 2006] (atoms of smaller size can be applied anywhere in the signal) or dictionary sparsity [Rubinstein et al., 2008b]: the dictionary itself is sparse on a meta-dictionary.

### 5.1.2 K-SVD

The K-SVD algorithm [Aharon et al., 2006], detailed in algo. 2 is based on a form of alternate optimization: first a sparse decomposition is performed on an initial dictionary, then the dictionary atoms are updated to fit better the signals in which they are used. The process is repeated a given number of times, or until (quasi-)convergence. Like many stat-of-the-art dictionary learning algorithm, it has a tendency to settle in local minima. This makes the algorithm very sensitive to the initialization, as a bad initialization can put the algorithm close to a local minimum that is very far from the global minimum.

**Sparse decomposition** Although this step can be implemented with any sparse decomposition algorithm, in our case it is implemented with BatchOMP, a fast algorithm for multiple OMP decompositions over the same dictionary. This algorithm precomputes the Gram matrix of the dictionary, which allows for faster computation on each signal. A more detailed explanation can be found in [Rubinstein et al., 2008a]

**Dictionary update** This step is performed one atom at a time, using only the frame examples that use this atom in their sparse representation. The atom is modified in order to fit the residual of the sparse approximation exempted of that atom. This update was originally performed with a SVD, hence the algorithm's name, however an approximate solution, such as the one provided by a single step of alternate optimization, is sufficient and much faster and is thus used instead.

**Algorithm 2** K-SVD [Aharon et al., 2006]

**input:** $\boldsymbol{D_0}, \boldsymbol{X}, Z, K$
**output:** $\boldsymbol{D}, \boldsymbol{\Gamma}$ such that $\boldsymbol{X} \approx \boldsymbol{D\Gamma}$
$\boldsymbol{D} := \boldsymbol{D_0}$
**for** n= 1..Z **do**
  $\forall i : \boldsymbol{\Gamma}_i := \underset{\underline{\gamma} \in \mathbb{R}^M}{\operatorname{argmin}} \left\| \boldsymbol{X}_i - \boldsymbol{D}\underline{\gamma} \right\|_2^2$ s.t. $\left\| \underline{\gamma} \right\|_0 < K$
  **for** k= 1..M **do**
    $I := \{i : \boldsymbol{\Gamma}_{k,i} \neq 0\}$
    **if** $I \neq \emptyset$ **then**
      $\boldsymbol{D}_k = \underline{0}$
      $\underline{g} := {\boldsymbol{\Gamma}_{I,k}}^T$
      $\boldsymbol{E} := {\boldsymbol{X}_I}^T - \boldsymbol{D\Gamma}_I$
      $\underline{d}, \underline{g} := \underset{\underline{d} \in \mathbb{R}^N, \underline{g} \in \mathbb{R}^{|I|}}{\operatorname{argmin}} \left\| \boldsymbol{E} - \underline{d}\underline{g}^T \right\|_2^2$ s.t. $\left\| \underline{d} \right\|_2 = 1$
      $\boldsymbol{D}_k = \underline{d}$
      $\boldsymbol{\Gamma}_{I,k} := \underline{g}^T$
    **else**
      replace $\boldsymbol{D}_k$ by a new atom
    **end if**
  **end for**
**end for**

## 5.2 Proposed algorithm: K-SVD with incomplete data

For inpainting applications, it can often occur that no clean data is available, for example if the whole track has been deteriorated. Straightforward learning of a dictionary on this kind of data will lead to propagating the corruption to the model. Applying the learned dictionary to corrupted data would then be useless or even counterproductive. We propose an adaptation of the K-SVD algorithm that is able to learn on locally distorted data without integrating the corruption to the dictionary, based on the assumption that the locations of the distorted samples can be determined beforehand. In order to avoid the propagation of the corruption, distorted samples are discarded and the learning process is only applied to the clean samples.

While conceptually simple, this discarding causes some issues with the way the K-SVD algorithm is computed, because the input vectors have different sizes and even when size is identical, the mapping to the full signal is different.

### 5.2.1 Sparse approximation step

The sparse approximation problem is the same as the one used in the inpainting method, thus there was an existing implementation of OMP that solves it. However, this implementation was costly to use in a dictionary learning algorithm such as K-SVD, that requires many computations of the sparse pursuit. BatchOMP, the algorithm used by the existing K-SVD, couldn't be applied directly because the dictionary is different for every distorted frame.

These dictionaries, although they are different, all come from a single dictionary and therefore have a common structure. One interesting property, coming from this structure, is that the Gram matrix $\boldsymbol{G}$ of the initial dictionary $\boldsymbol{D}$ can be be written as the sum of the Gram matrices $\boldsymbol{G}^r$ and $\boldsymbol{G}^m$ of $\boldsymbol{D}^r$ and $\boldsymbol{D}^m$, the dictionary restricted to the locations of respectively the clean samples and the distorted samples (demonstration in annex A). The number of distorted samples is typically much lower than the number of clean samples, therefore computing $\boldsymbol{G}^m$ then $\boldsymbol{G}^r$ as the difference between $\boldsymbol{G}$ and $\boldsymbol{G}^m$ is less costly than computing it directly from $\boldsymbol{D}^r$. BatchOMP can then be used with the adapted Gram matrix.

Experimentally, this setup ends up being about 30% faster than the OMP-Cholesky implementation available along BatchOMP with the parameters that were used during the experiments. This speedup could be enhanced by performing the Gram update in C++ (like the BatchOMP and OMP-Cholesky implementations) instead of matlab, but by lack of time, it was not.

### 5.2.2 Update step

The aim of the update step remains the same as in the standard version of K-SVD: each base atom is successively optimized to represent best the frames it is used in, when used with the other atoms in the dictionary. However, the actual atom used may differ from the base atom because it had some samples removed by the decimation matrix. Moreover, which samples are missing depends on the frame.

Even then, this problem can still be expressed as the least squares solution of a linear equation system (*cf.* annex B). In most cases, the solution is unique, but when it is not it usually means that the atom is used by very few frames and some samples are corrupted in every frame that uses the atom. In that case, the atom could be replaced with another one that would be used by more frames. Another solution, which we implemented, is to consider the free variables in the solution as missing samples and inpaint the atom accordingly in order to get a full atom.

## 5.3 Investigation axes

**Algorithm 3** K-SVD with incomplete data

**input:** $\boldsymbol{D_0}, \boldsymbol{X}, \boldsymbol{R}, Z, K$
**output:** $\boldsymbol{D}, \boldsymbol{\Gamma}$ such that $\boldsymbol{R} \circ \boldsymbol{X} \approx \boldsymbol{R} \circ \boldsymbol{D\Gamma}$
$\boldsymbol{D} := \boldsymbol{D_0}$
**for** n= 1..Z **do**
  $\forall i : \boldsymbol{\Gamma}_i := \underset{\underline{\gamma} \in \mathbb{R}^M}{\operatorname{argmin}} \left\| \boldsymbol{R_i} \circ \left( \boldsymbol{X}_i - \boldsymbol{D}\underline{\gamma} \right) \right\|_2^2$ s.t. $\left\| \underline{\gamma} \right\|_0 < K$
  **for** k= 1..M **do**
    $I := \{ i : \boldsymbol{\Gamma}_{k,i} \neq 0 \}$
    **if** $I \neq \emptyset$ **then**
      $\boldsymbol{D}_k = \underline{0}$
      $\underline{g} := {\boldsymbol{\Gamma}_{I,k}}^T$
      $\boldsymbol{E} := {\boldsymbol{X}_I}^T - \boldsymbol{D\Gamma}_I$
      $\underline{d} := \underset{\underline{d} \in \mathbb{R}^N}{\operatorname{argmin}} \left\| \boldsymbol{R}_I \circ \left( \boldsymbol{E} - \underline{d}\underline{g}^T \right) \right\|_2^2$ s.t. $\left\| \underline{d} \right\|_2 = 1$
      $\underline{g} := \underset{\underline{g} \in \mathbb{R}^{|I|}}{\operatorname{argmin}} \left\| \boldsymbol{R}_I \circ \left( \boldsymbol{E} - \underline{d}\underline{g}^T \right) \right\|_2^2$
      $\boldsymbol{D}_k = \underline{d}$
      $\boldsymbol{\Gamma}_{I,k} := \underline{g}^T$
    **end if**
  **end for**
**end for**

# 6 Experiments

In this section, we will describe the experimental system and the experiments that we conducted in order to:

1. establish a baseline system to compare the learning algorithms with,
2. find favorable setups for dictionary learning with K-SVD,
3. and evaluate the new algorithm we proposed in section 3.

## 6.1 Experimental system

Figure 4 presents an overview if the declipping system used for the experiments. We detail below the steps that appear in this figure.

**Synthetic clipping** We did not have data that was available in both clipped (to use as input) and clean (to compare the estimate with the groundtruth) forms. Thus, we applied a synthetic clipping to clean signals. This clipping might differ slightly from real-life clipping because it is a hard clipping, meaning that sample values above the clipping level are mapped to a single value, the clipping level (the same applies for negative samples below the clipping level); real-life clipping may sometimes be soft clipping, *i.e.* the values that exceed the clipping level are distorted, but an order relation is kept. Figure 5 shows transfer functions for both hard and soft clipping. Soft clipping is harder to detect, however since the transfer function is bijective, it is easier to reverse,

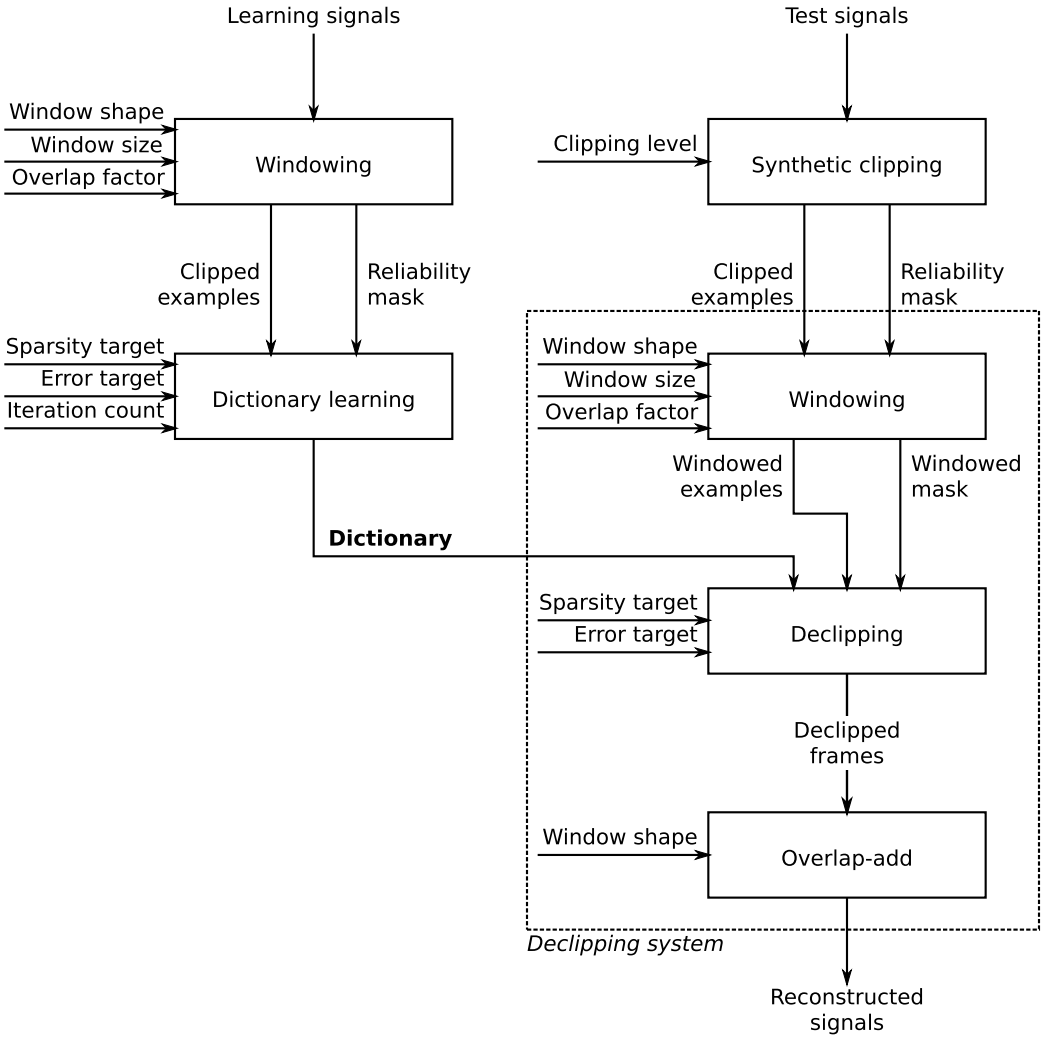


Figure 4: Declipping workflow (top to bottom, parameters on the left)

provided that the transfer function can be inferred from training data – we do not treat that case. For real applications of the declipping, this step is replaced by a clipping detection step, that can then pass the reliability mask along to the rest of the system.

**Windowing and Overlap-add** Most sparse decomposition algorithms have a complexity that does not allow them to be applied to a full signal. They have to work on short frames of data in order to keep tractable computation times. Therefore, we split the signal in overlapping frames using a windowing process

$$\boldsymbol{X}_{i,j} = w_a\left[j\right] \times \underline{x}\left[u(i) + j\right] \tag{10}$$

where $w_a$ is the window discrete function and $u(i)$ is the beginning offset for frame $i$.

The overlap-add step corresponds to the inverse operation: going back from the frames to a full signal. It uses a synthesis window $w_s$ which is reweighted according to the overlap factor and

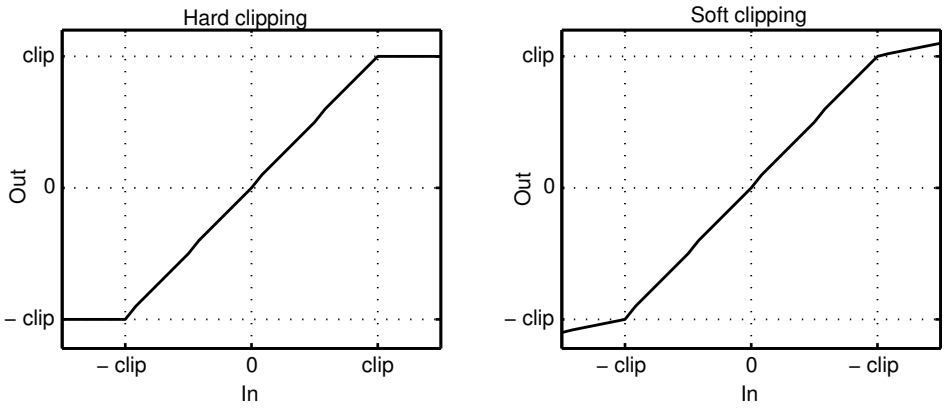


Figure 5: Transfer functions for hard (left) and soft (right) clipping

the analysis window so that a signal going through only the windowing and overlap-add steps is unchanged.

**Declipping** This is the main step, where the input frame is transformed in its sparse approximation. The sparse approximation is then used to build an estimate of the signal. The most important input for this step in the frame of our work is the dictionary: this is where the dictionary learning algorithms' output is plugged in.

**Dictionary learning** The dictionary learning involves the same step of windowing so that the learning signals resemble the test signals, but not the overlap-add since no signal synthesis is needed. The different signals of the learning set are appended to one another after the windowing, in order to have a single input matrix for the learning. The learned dictionary is then passed on to the main declipping system.

**Bootstrapping** Figure 6 shows the modified workflow for the bootstrapping application. No training set is used, as the learning is performed directly on the clipped samples. Instead, the input of the dictionary learning algorithm comes from the artificial clipping.

## 6.2 Tools

### 6.2.1 Matlab

Matlab is a commercial software for scientific computation developed by Mathworks. It revolves around the use of scripts and functions in its own language. Matlab's main assets are the many built-in functions and additional toolboxes for targeted applications (such as signal processing, neural networks or virtual reality), which allow for quick implementation of new or existing algorithms. Its debugger and plotting tools are also great features.

**ksvdBox and ompBox** The ksvdBox and ompBox toolboxes were released following the work presented in [Rubinstein et al., 2008a] and include the OMP-Cholesky, BatchOMP and K-SVD algorithms.

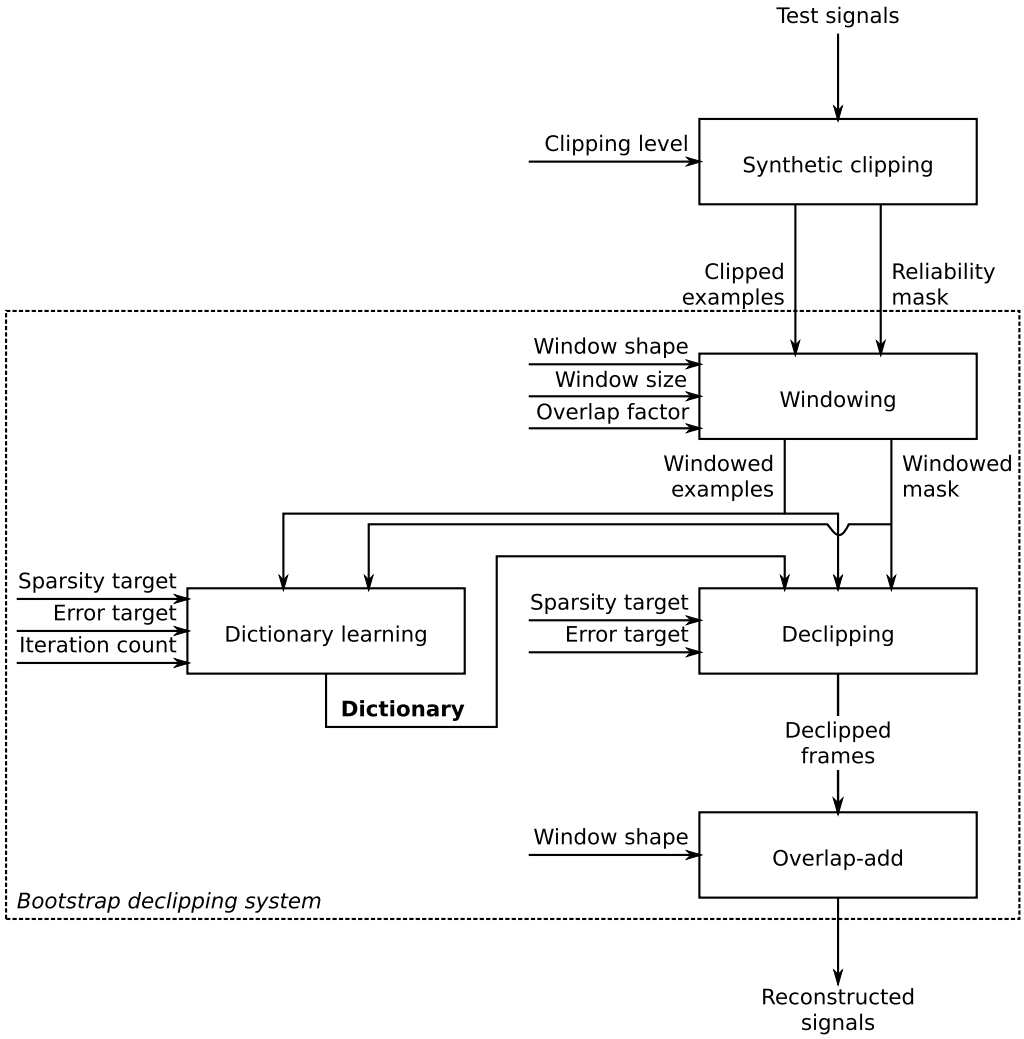


Figure 6: Bootstrapping modified workflow

**Audio Inpainting Toolbox** The audio inpainting toolbox is a free implementation of the work this internship is based upon. It includes the necessary code to reproduce the experiments in [Adler et al., 2011], and design new ones.

Those toolboxes are available as parts of the SMALLbox[4]. They required a familiarization period before I could use and understand them completely.

### 6.2.2 Computation cluster: Igrida

Early tests were performed locally, but as the length and number of executions grew, it has quickly become mandatory to use the Inria/Irisa cluster: Igrida. This cluster is running with the batch scheduler Sun Grid Engine (SGE). However, it was not possible to run matlab code directly,

[4] `small-project.eu/software-data/smallbox`

because each run requires a license for its whole duration, preventing many simultaneous runs and possibly blocking other matlab users at the research center. The code had to be compiled to a standalone application, which required some modifications, including the replacement of the library used for the constrained optimization involved in the declipping specialized version of OMP. The matlab compiler requires a specific license which was occasionally unavailable, implying some delays in the experimental process. Some unresolved issues also occasionally caused the compilation to fail for undetermined reasons, without blocking the execution, causing further delays. However, the delays that were induced by the use of the cluster were more than compensated by the ability to run several tests at once.

## 6.3 Methodology

We decided to experiment in 3 stages: first, we attempted to establish a strong baseline through careful parametrization of the existing system of the audio inpainting toolbox; we then tested the influence of learning a dictionary with the K-SVD algorithm in different situations, and finally we tested our modified version of K-SVD.

### 6.3.1 Datasets

All signals are normalized to a maximum absolute amplitude of 1, in order for the clipping to be comparable. To the exception of the first dataset, all were generated during the internship for the needs of our experiments.

**Speech** A set of ten 5-second signals from the 2008 Signal Separation Evaluation Campaign [Vincent et al., 2009] that was used in [Adler et al., 2011], including different speakers (male and female) in different languages, sampled at 16kHz. However, in order to get a more homogeneous dataset for learning, we kept only the 5 male speakers, as well as an extra excerpt which met the same recording conditions.

**MusicShort** 5-second excerpts from the first 20 songs (2 per song, at different locations in the song) of the RWC_POP music database [Goto et al., 2002], consisting in Japanese popular music. The signals were downsampled from 44.1kHz by a factor of 3, in order to have sampling rates similar to the speech datasets.

**MusicLong** Similar to the MusicShort dataset, but with 1-minute excerpts. There is no overlap between excerpts of the same song.

**SpeechGripari** Twenty 5-second excerpts from a children tale ("*La Paire de Chaussures*", told by Pierre Gripari). The initial file was encoded with Apple MPEG-4 and was converted to 16kHz waveform for the experiments. This set contains plain speech with occasional sound effects (bells, short musics); the excerpts were taken so that they were not back to back, and that training and test excerpts were interlaced. However, as of this writing, we did not have the time to experiment on this dataset.

### 6.3.2 Performance measurement

The quality of the reconstruction was measured with 2 slightly different measures, the signal-to noise ratio (SNR) defined by

$$\mathrm{SNR}(\underline{y}, \underline{\hat{y}}) = 10 \log \frac{\left\|\underline{y}\right\|_2^2}{\left\|\underline{y} - \underline{\hat{y}}\right\|_2^2} \tag{11}$$

and a derived value, noted $\mathrm{SNR}_m$, corresponding to the SNR computed on distorted samples and defined by

$$\mathrm{SNR}_m(\underline{y}, \underline{\hat{y}}) = 10 \log \frac{\left\|\underline{y}^m\right\|_2^2}{\left\|\underline{y}^m - \underline{\hat{y}}^m\right\|_2^2} \tag{12}$$

and related to SNR by

$$\mathrm{SNR}(\underline{y}, \underline{\hat{y}}) = \mathrm{SNR}_m(\underline{y}, \underline{\hat{y}}) + 10 \log \frac{\left\|\underline{y}\right\|_2^2}{\left\|\underline{y}^m\right\|_2^2}. \tag{13}$$

SNR reflects the end quality of the reconstructed signal, while $\mathrm{SNR}_m$ reflects how well the missing samples were reconstructed. The use of these two measures is discussed in the next subsection.

### 6.3.3 Issues with measuring declipping performance

Clipping is an easy to reproduce, yet hard to control process. Since the location and number of clipped samples depends on the signal, the same clipping level will produce declipping problems of different difficulties. Different properties of the clipping have to be considered:

**Clipping level** The ratio between the highest absolute value of the source signal and the value of clipped samples. Its value cannot be known with certainty in a real case, however it can be estimated, for example using a MAP approach.

**Clipped-to-clean ratio** Estimating more based on less is of course harder.

**Clipped samples distribution** Experiments from [Adler et al., 2011] showed that for a fixed missing-to-clean ratio (fixed signal length and number of missing samples), the reconstruction is easier when distortions are shorter but more numerous.

Those 3 properties are closely tied, and the only one that can be controlled directly (in a synthetic clipping) is the clipping level. The traditional measure of SNR is very sensitive to the clipped-to-clean ratio: since it involves clean samples on which no reconstruction is needed, the value is shifted upwards independently from the inpainting process.

In order to counteract this shift, we use $\mathrm{SNR}_m$. This measure is more stable regarding the clipped-to-clean ratio: a constant error per clipped sample will result in an almost constant $\mathrm{SNR}_m$ (except for very few clipped samples, where the signal energy is unstable). Figure 7 shows the two measures on a clipped speech excerpt depending on clipping level. Since the difficulty of the problem increases with the number of clipped samples, the error per clipped sample is not constant, which results in different signals sometimes having quite different $\mathrm{SNR}_m$ from one another (up to 10dB, less different than SNRs). Fortunately, the variations along the inpainting parameters do not differ much between signals, meaning that comparisons are possible on a global scale (although this does not hold for variations of the dictionary).

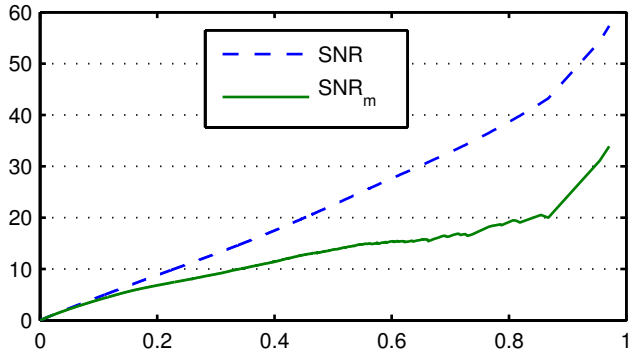


Figure 7: SNR and $\text{SNR}_m$ of the clipped signal at different clipping levels

Another workaround would have been to set the clipping level in order to reach a fixed clipped-to-clean ratio. This other approach would also have reduced the variance of the problem difficulty, however we did not explore it. Normalizing the signals according to mean energy instead of maximum absolute amplitude could also have led to different results.

It is important to note that those measures have a limited relation to perceived quality: an estimate having a high SNR can sound worse than another with a lower SNR. However, they still show good correlation, and they are much simpler to compute than perceptual measures like PEAQ [PEA, 1998] that take human audition models into account when computing a value.

#### 6.3.4 Comparison with state-of-the art methods

The base inpainting system that we are using has been shown to achieve state-of-the-art results. Therefore we designed our experiments to use this base version as a reference.

### 6.4 Baseline

In order for our comparison with unlearned dictionaries to be relevant, we first had to find some good parameters for the inpainting process. Because of the number of parameters to set is too high to optimize for all parameters at once, we proceeded in successive steps of parameter exploration for a few related parameters (window shape and size for example). We attempted to choose the order of these steps so that optimal parameters chosen at early steps do not change much during the later steps, using reasonable (based on the experiments in [Adler et al., 2011] and informal initial experiments) settings for the parameters that were yet to optimize. Even then, the experiments took a considerable amount of time and had to run during the weekend.

#### 6.4.1 Windowing process

Because of complexity issues, we cannot inpaint the signal as a whole. We decided to study several parameters for the windowing process: the size of the window, the shapes of both the analysis and synthesis windows, and the overlap factor, *i.e.* how many different windows use a single sample.

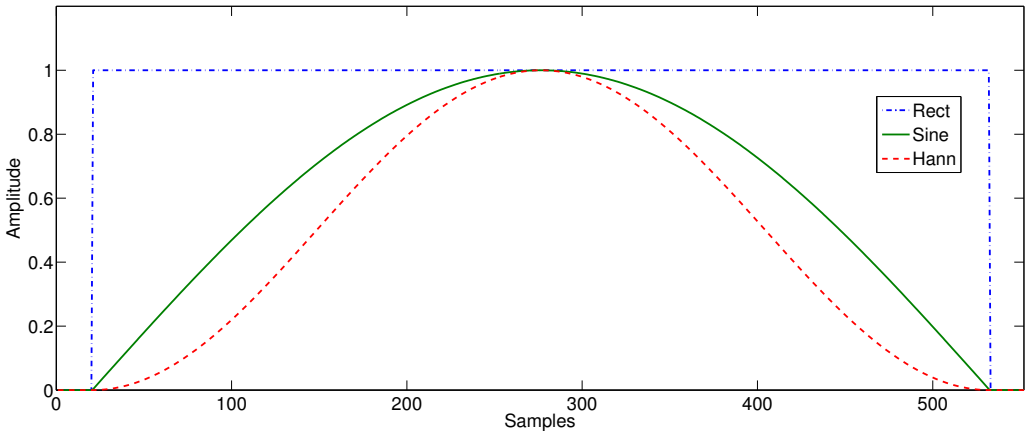


Figure 8: The rectangular, sine and Hann windows

**Window shapes** We tested for 3 different window shapes: rectangular, sine and Hann (squared sine), shown in Fig. 8. The reweighting of the synthesis windows allows for any combination of windows to be used, therefore we tested every combination of these 3 analysis and synthesis shapes.

Figure 9 shows average $\text{SNR}_m$ over the 6 speech excerpts with a window size of 512 samples and an overlap factor of 2. We also did this experiment with double window length and double overlap factor, but these values are not shown to prevent a cluttered figure. The results show the same tendencies with other the window lengths and overlap factors. We can see that the analysis window has a lot more influence than the synthesis window, and that the sine shape obtains better results than the other ones for analysis. The results are much closer for synthesis shapes; on average over the 4 setups the sine window performs on par with the Hann window and slightly better (+0.2dB) that the rectangular window.

We chose to move on with a sine-sine scheme for the following experiments because of the good results of the sine window for analysis and the fact that the normalization step is not required for this pair.

**Window size and overlap** From our initial window size of 512 and 50% overlap, we tried doubling the window size and the overlap factor. Both showed an improvement between 0.5 and 2dB, depending on the clipping level. We tried increasing them further (the results are presented in fig 10) but the yield was not as good and not worth the computing time increase (2 to 4 times longer) anymore in our opinion. For similar improvements, increasing the overlapping turned out to be slightly less costly.

### 6.4.2 Sparse decomposition

**Dictionary selection** We tried using an union of DCT bases (blocks) at different scales and a Gabor dictionary. However, the implementation of OMP with the Gabor dictionary was much more costly (40x) than with the UDCT. Thus, even though better results (+1-2dB) were reported

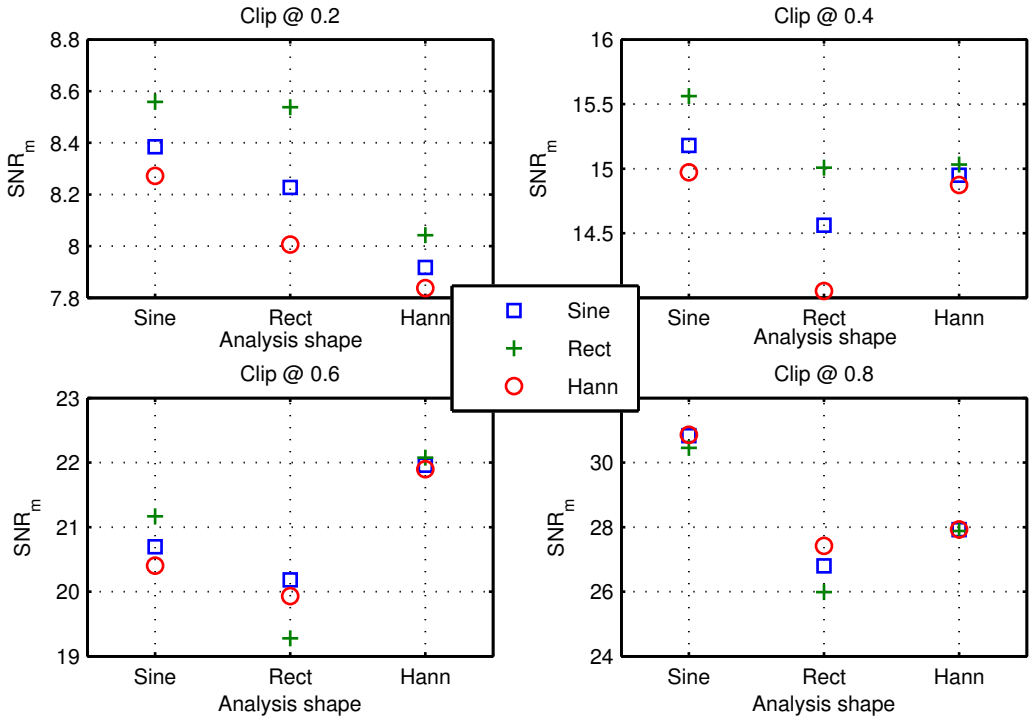


Figure 9: $\mathrm{SNR}_m$ for different combinations of window shapes. Marks are for the synthesis window.

in [Adler et al., 2011]using a Gabor dictionary, we ruled it out until a more efficient implementation is available. We then experimented on the UDCT, modifying the number of scales and the frequency redundancy of the blocks. The best results we obtained were with no redundancy in the blocks and either 1 or 2 scale levels. We kept the union with 2 levels, in spite of slightly inferior results, in order to retain a redundant dictionary.

**Stopping criterion** Previous experiments [Adler et al., 2011] revealed that the algorithm was quite sensitive to the stopping criterion. Thus, we tried to fine-tune these parameters. We tested for sparsity target ranging from $2^4$ to $2^9$ (for a 1024-sample-long window) and squared error from $10^0$ to $10^{-4}$. The error criterion turned out to be either too loose, resulting in much lower $\mathrm{SNR}_m$ values, or too strong, resulting in sparsity being the stopping condition for OMP. Figure 11 shows results for $\epsilon$ low enough to go through $K$ iterations, at various clipping levels.

We could not find a direct relation between the best number of iterations and the clipping level. However, the $\mathrm{SNR}_m$ plateaus after N/8 for low clipping levels, hinting that harshly clipped signals have a lower optimal sparsity target. Further experiments will focus on relating the optimal target sparsity to the number and distribution of missing samples on a frame-by-frame basis. Indeed, even with harsh clipping, a lot of frames can have few missing samples, weighting down a low sparsity target.

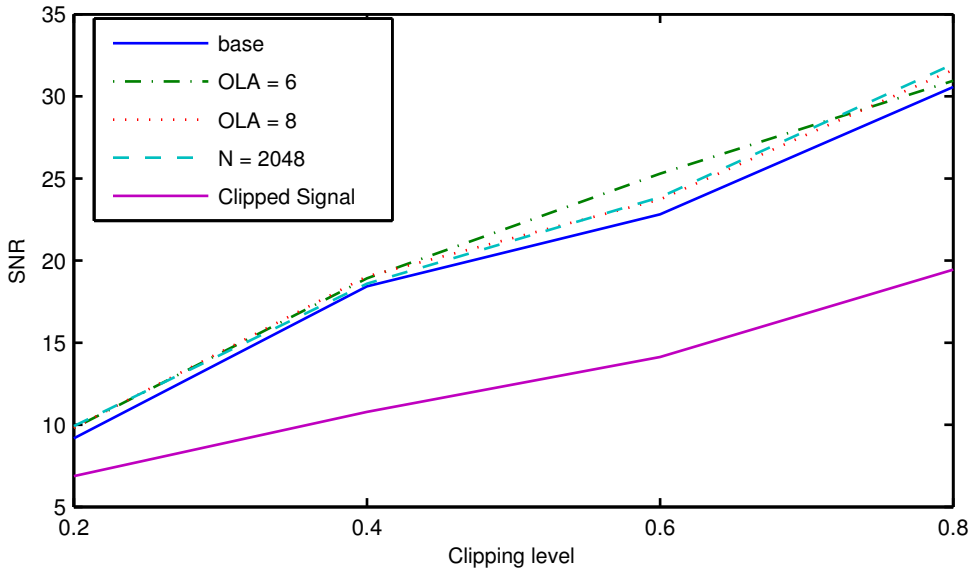


Figure 10: SNR vs. clipping level for higher window size and overlap factors. Base corresponds to $N = 1024$ with an overlap factor of 4.

| | |
|---|---|
| Frame length | 1024 |
| Analysis window shape | Sine |
| Synthesis window shape | Sine |
| Overlap | 75% |
| Dictionary | Union of DCT (2 scales) |
| OMP iterations | 512 |

Table 1: Retained parameters for the remaining experiments (baseline system)

**Baseline system** Table 1 sums up the parameters we retained for our baseline system. In order for a system using a dictionary learning algorithm to be considered useful in our application, the learning system has to perform better than this baseline. Most of the parameters we determined also apply to learned dictionaries; these parameters are a reasonable setting as we expect their behavior to be similar with different dictionary and the exploration of the parameter space is much more costly when the dictionary has to be learned.

### 6.5 K-SVD dictionary learning

**Training procedure** From the training dataset, we generated a set of frames to feed to the dictionary learning. This set has a greater overlap than the set of frames built in the declipping process (the hop size is 64 instead of 256), in order to increase the number of training frames and ensure that we have enough data to train our atoms. We initialized the algorithm with our baseline dictionary, an union of DCT bases (UDCT).

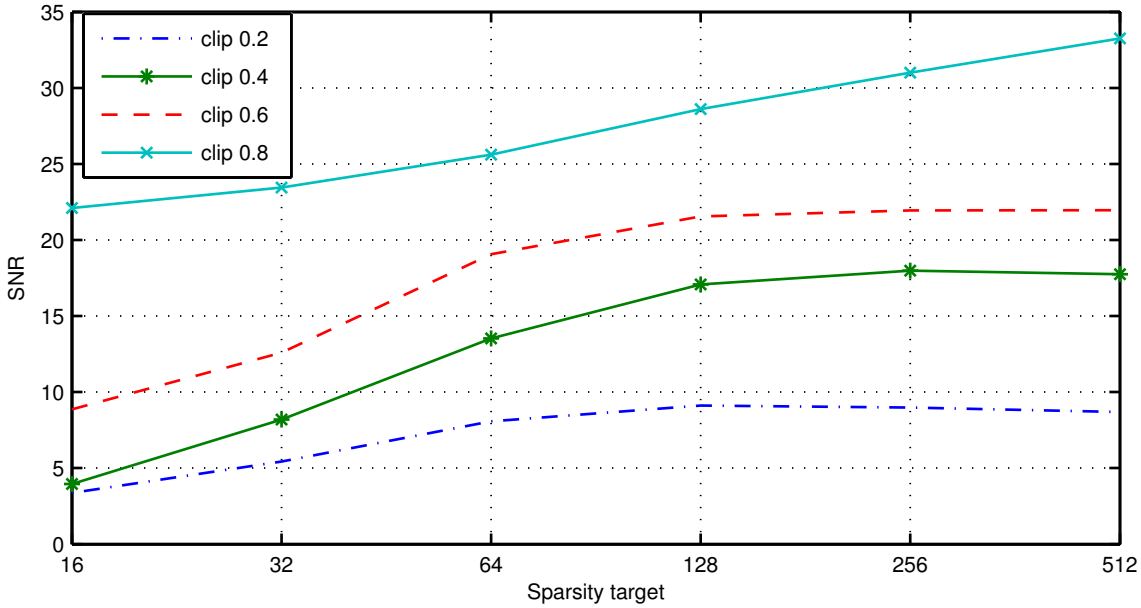


Figure 11: $\text{SNR}_m$ vs. sparsity target at different clipping levels

Learning a dictionary is computationally heavy task. Therefore we tried to maximize the use we could make out of the dictionaries once they were computed, learning as few new dictionaries as possible.

**Speech experiment (small dataset)** We started our experiments on K-SVD with a simple setup, with 4 signals of the Speech set as learning set, testing on the remaining 2. We decided to try several values of $K$, because we expected the learned dictionary to cope much better with a low sparsity target than the pre-established one and perhaps obtain better results with few atoms than with many. Each dictionary is trained with the sparsity target it is later applied with. We chose to test for each value of $K$ the first and the last iteration.

Figure 12 shows the results of this experiment. The learned dictionaries do indeed perform better than the UDCT with very sparse approximations, however this performance is a degradation compared to the clipped signal according to SNR. Furthermore, we expected the learned dictionary to perform better than the UDCT overall. Against our expectations, the learned dictionaries obtained slightly worse to significantly worse results, that seem to deteriorate with the number of K-SVD iterations.

**MusicShort nominal experiment** Our first hypothesis for the reason of K-SVD's failure to improve decliping results was that 4 samples of 5s, even with an overlap factor of 16, was too small a training set to learn a good dictionary. We decided to change our dataset to a bigger one, MusicShort ($2 \times 20 \times 5$s). We split the base in training and test sets so that two excerpts from the same song would not appear in both datasets. We also decided to discard very low values of $K$ (¡64), that yielded poor results in the previous experiment, to reduce the number of runs to execute. The dictionary learning took longer than anticipated for the highest value of $K$ and hit

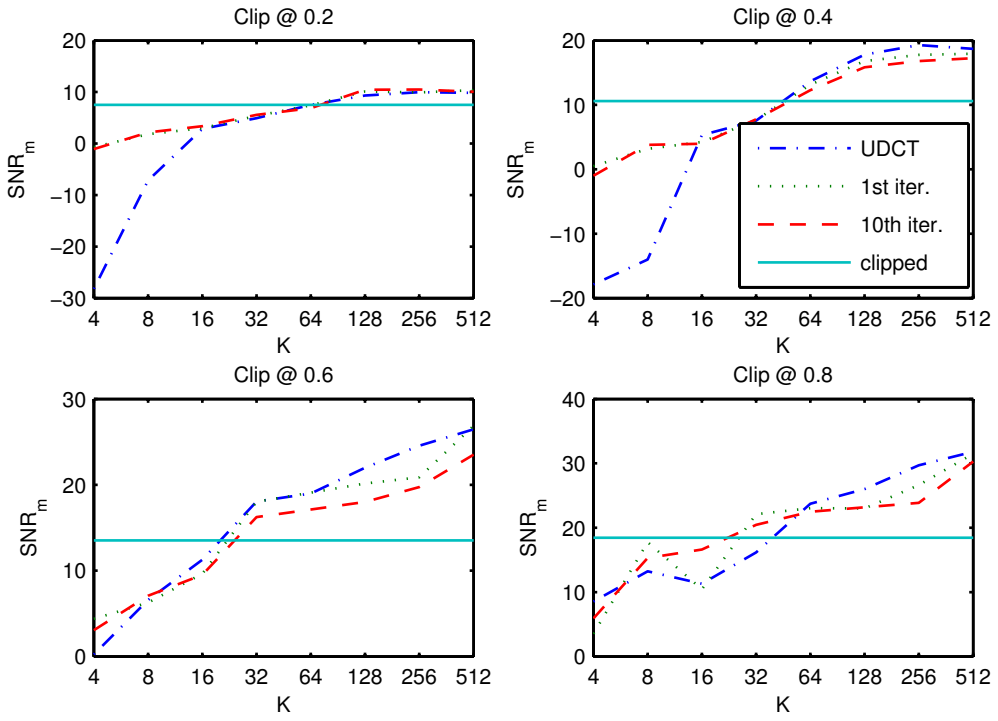


Figure 12: Declipping performance for three dictionary series at different sparsity target.

the allocated time-limit on the batch scheduler. We still performed the tests with 4 iterations of K-SVD since each iteration is supposed to be an upgrade anyway.

Results are presented in fig 13. The mean difference from one iteration to the next is slightly negative. However, for difference as small as half a dB between two signal, informal listening showed that the estimate built with the learned dictionary sounded very noisy.

**Single song experiment** We considered the diversity of music (even within a genre) as another possible source for the poor performance of K-SVD in the previous experiments. To reduce this diversity and confirm or infirm this hypothesis, we changed the way we extracted the excerpts from the RWC_POP database, in order to have one excerpt, long enough to compare with our previous dictionary learning experiments, from the beginning of the song, and another excerpt from the end of the song to test the learned dictionary on.We repeated that process for the 20 songs, and dropped the remaining values of $K$, keeping only 512, to avoid learning too many dictionaries, and limiting the experiment to a 0.4 clipping level; the results are exposed song by song in fig. 14. We can see that after the first iteration, very little changes occur in the SNR.

**MusicShort training subset experiment** At that point, we wanted to check that K-SVD was actually learning something from the signals. We applied the learned dictionaries to the training samples, without clipping. The set of frames that was processed was therefore a subset (1 out of 4) of the set used to learn the dictionary. Fig. 15 shows that the SNR of signals oscillate strongly for independent signals. Even though this SNR is quite high, this seems to indicate again that the K-SVD algorithm is stuck near a local optimum.

A last minute investigation revealed that learned atoms and by extension the dictionaries, are very similar to their initialization (see fig 16). It looks like K-SVD changed very little but adding what appears to be noise instead of silence. This is clearly a sign of a local optimum. Finding a way to get past that local optimum will be the focus of the remaining month of the internship.

### 6.6 K-SVD with incomplete data

**Speech experiment (small dataset)** We ran some preliminary experiments on the Speech datasets to test the implementation of the modified K-SVD algorithm. Its performance was just below the regular K-SVD , which is to be expected since we discard some data for this algorithm. Unfortunately, we lacked the time to run more experiments, including the main use case of bootstrapping.

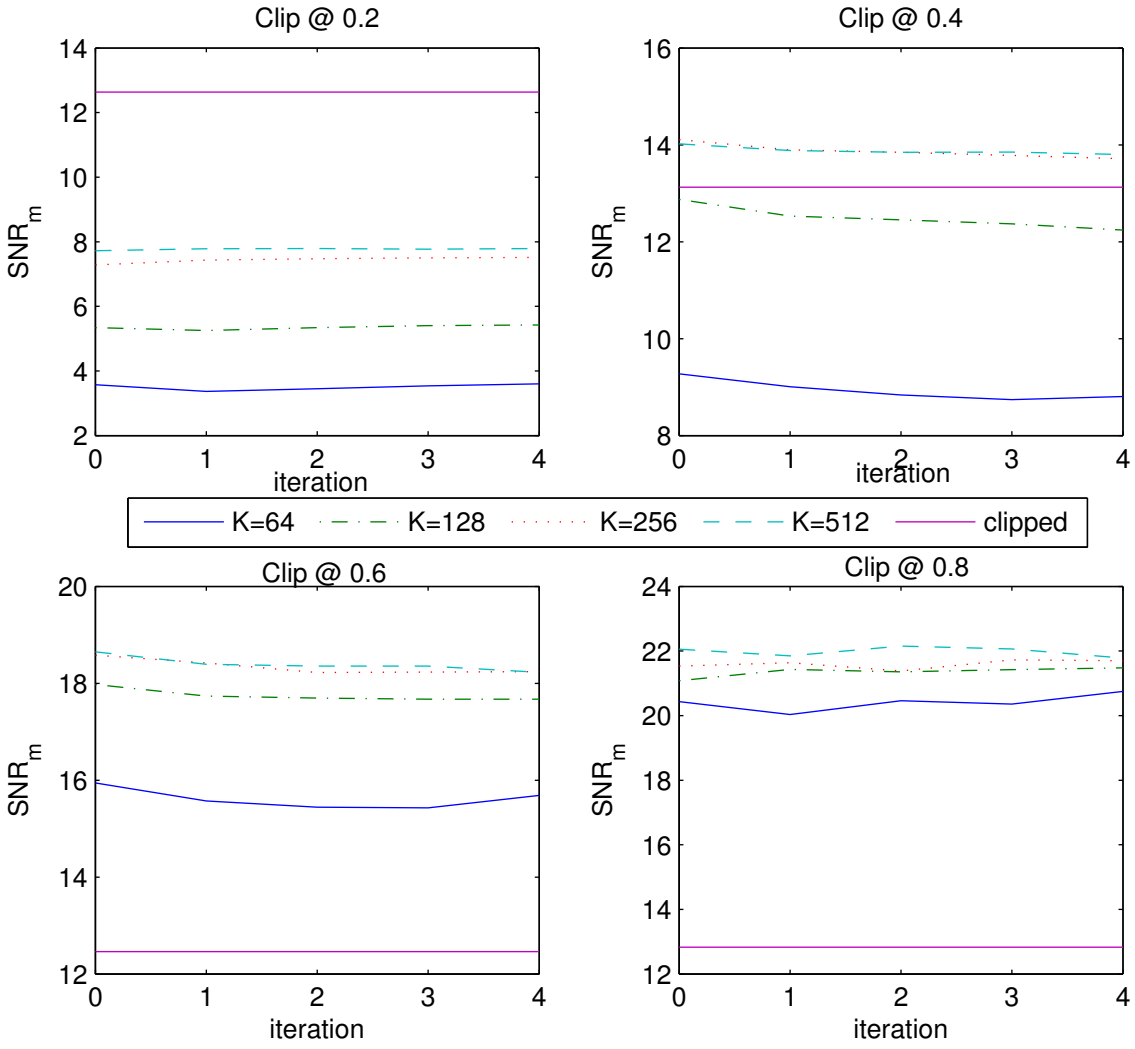


Figure 13: Nominal MusicShort experiment

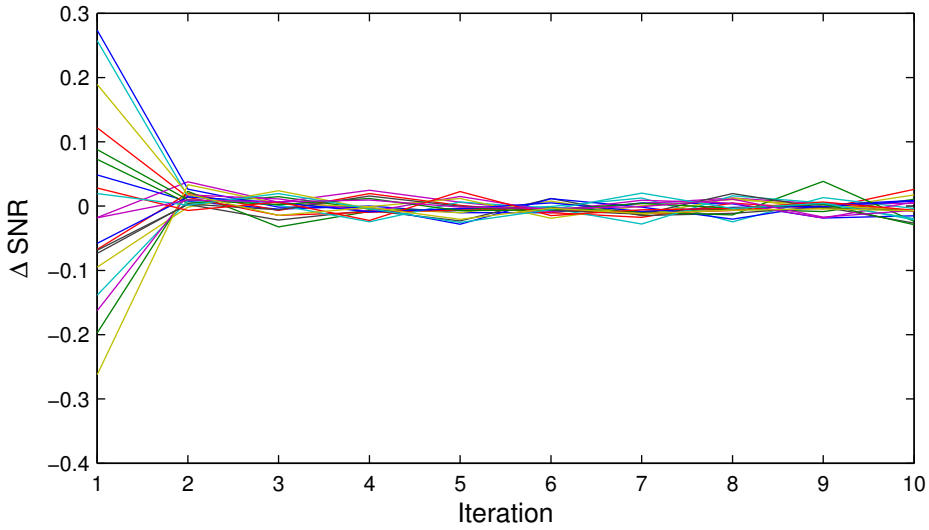


Figure 14: SNR difference from one dictionary to the next one

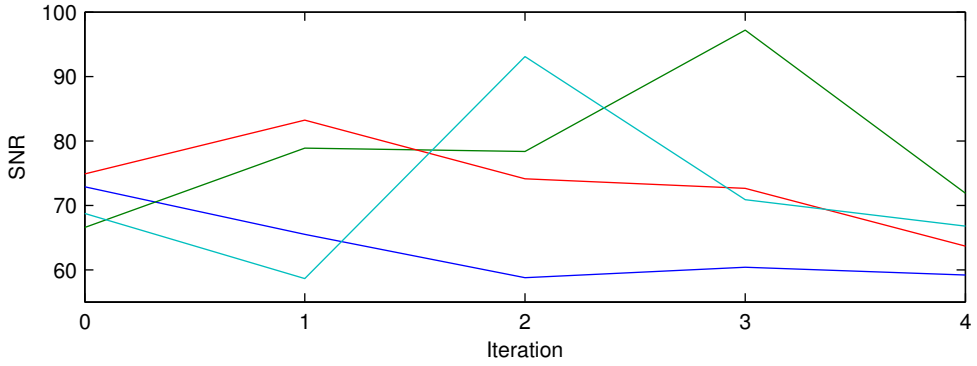


Figure 15: Training subset experiment

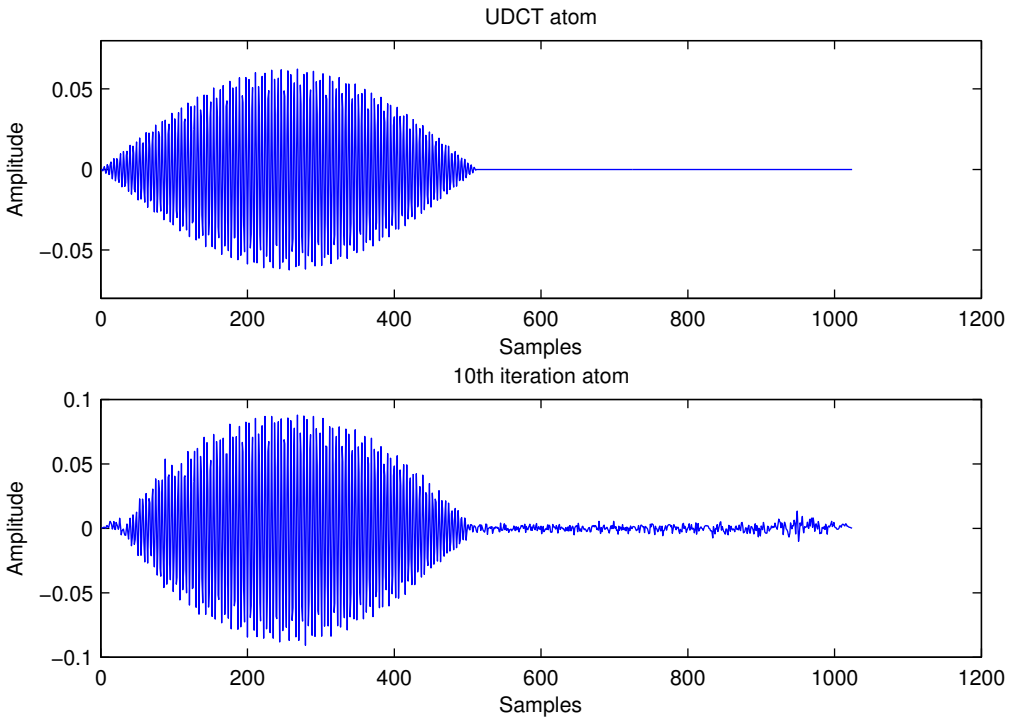


Figure 16: Initial and learned atoms

# 7 Work in progress and perspectives

Since this report is being submitted almost one month before the end of the internship, additional work will be realized. This section presents the possible developments of this last month, and envisioned axes for further research.

## 7.1 Initialization

As it seems that our experiments encountered a local minimum with the UDCT dictionary, we want to explore options to get away from that local minima. In the next few weeks, we plan to experiment with the initialization: we would like to try random initializations and check if the algorithm manages to improve this most likely poor dictionary. We would also like to test an initialization with frames picked out of the training signals, in the hope that these frames are representative of the whole signal.

## 7.2 Bootstrapping

We also plan to experiment with bootstrapping of the inpainting process using our proposed algorithm. If those experiments were conclusive, the approach would be applicable in real case scenarios, without requiring any prior knowledge on the signal to restore.

## 7.3 Sparsity target adaptation to frame

Following our experiments and the ones presented in [Adler et al., 2011] on the stopping criterion of the decomposition algorithm, we would like to search for a relation on the frame level between the severity of the degradation and the optimal criterion setting. Such a relation, even roughly approximated, would allow for speedups (since we currently use a high sparsity target that could be set lower), and better results by avoiding the modeling of the noise that can still be present in the clean data.

## 7.4 Shift invariance

A lead for further work is the use of algorithms that focus on translation-invariance. Indeed, the physical phenomena that produce the sounds do not fit in the same part of the time frame at every occurrence. We are considering two leads for the use of time-invariance.

First, we want to evaluate the efficiency of decomposing the whole signal at once (skipping the windowing process) over a translation invariant dictionary. Such a dictionary could for example be built by collecting all translations of the atoms of one of the dictionaries we used for our experiments. Thanks to efficient implementations such as the one proposed in MPTK [Krstulovic and Gribonval, 2006], this could even translate to a speedup of the inpainting method.

In a second time, if the first experiments on translation-invariance are a success, we would like to learn translation invariant dictionaries to enhance the quality of the inpainting, and possibly try to adapt another dictionary learning to the bootstrapping scheme.

# 8 Conclusion

During this internship, I focused on reversing the distortions that occur locally in audio signal, a task recently referred to as *audio inpainting*. In order to achieve this, I have used existing tools and algorithms that make use of *sparse modeling*, stating that signals can be expressed as combinations of a few elementary pieces of signal called *atoms*. This model takes as its main parameter a large bank of atoms, the *dictionary*. The choice of this dictionary is often made based on prior knowledge of generative model of the signal, but a growing approach is to use *machine learning* principles to be able to adapt the dictionary to the signal.

I experimented in the framework of audio inpainting with K-SVD, a well-known algorithm to perform dictionary learning. I also proposed a modification of this algorithm for a concrete application where no training data is available. With my experiments, I realized that using machine learning does not necessarily improve the result, and I identified a major issue caused by some of the choices I made. In the rest of this internship, I will focus on resolving this issue and then on the validation of my proposed modification.

# A Gram update

$$\boldsymbol{G} = \boldsymbol{D}^T \boldsymbol{D} \tag{14}$$

$$\boldsymbol{D} = \boldsymbol{M}_f^r \boldsymbol{D} + \boldsymbol{M}_f^m \boldsymbol{D} \tag{15}$$

We denote by $\boldsymbol{M}_f^r$ (resp. $\boldsymbol{M}_f^m$) the square decimation matrix keeping only reliable (resp. missing) samples and nullifying the others, *i.e.* the matrix obtained by taking the identity matrix $\boldsymbol{I}$ and setting $\boldsymbol{I}_{j,j} := 0$ when sample $j$ is missing (resp. reliable).

$$\boldsymbol{M}_f^r\,(i,j) = \begin{cases} 1 & \text{if } i = j \text{ and sample } j \text{ is reliable} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

$$\begin{aligned} \boldsymbol{G} = &\left(\boldsymbol{M}_f^r \boldsymbol{D}\right)^T \boldsymbol{M}_f^r \boldsymbol{D} + \left(\boldsymbol{M}_f^m \boldsymbol{D}\right)^T \boldsymbol{M}_f^m \boldsymbol{D} \\ &+ \left(\boldsymbol{M}_f^r \boldsymbol{D}\right)^T \boldsymbol{M}_f^m \boldsymbol{D} + \left(\boldsymbol{M}_f^m \boldsymbol{D}\right)^T \boldsymbol{M}_f^r \boldsymbol{D} \end{aligned} \tag{17}$$

$$\begin{aligned} \boldsymbol{G} = &\boldsymbol{D}^T \boldsymbol{M}_f^r \boldsymbol{M}_f^r \boldsymbol{D} + \boldsymbol{D}^T \boldsymbol{M}_f^m \boldsymbol{M}_f^m \boldsymbol{D} \\ &+ \boldsymbol{D}^T \boldsymbol{M}_f^r \boldsymbol{M}_f^m \boldsymbol{D} + \boldsymbol{D}^T \boldsymbol{M}_f^m \boldsymbol{M}_f^r \boldsymbol{D} \end{aligned} \tag{18}$$

By construction, $\boldsymbol{M}_f^m \boldsymbol{M}_f^r = \boldsymbol{M}_f^r \boldsymbol{M}_f^m = \boldsymbol{0}$. One can easily verify that $\boldsymbol{M}_f^r \boldsymbol{M}_f^r = \boldsymbol{M}_f^r = \boldsymbol{M}^{rT} \boldsymbol{M}^r$

$$\tag{19}$$

$$\boldsymbol{G} = \boldsymbol{D}^T \boldsymbol{M}^{rT} \boldsymbol{M}^r \boldsymbol{D} + \boldsymbol{D}^T \boldsymbol{M}^{mT} \boldsymbol{M}^m \boldsymbol{D} \tag{20}$$

$$\boldsymbol{G} = \boldsymbol{D}^{rT} \boldsymbol{D}^r + \boldsymbol{D}^{mT} \boldsymbol{D}^m \tag{21}$$

$$\boldsymbol{G}^r = \boldsymbol{G} - \boldsymbol{G}^m \tag{22}$$

# B Modified dictionary update

$$\underline{d} = \underset{\underline{d} \in \mathbb{R}^N}{\operatorname{argmin}} \left\| \boldsymbol{R}_I \circ \left(\boldsymbol{E} - \underline{d}\underline{g}^T\right) \right\|_2^2 \tag{23}$$

$$\underline{d} = \underset{\underline{d} \in \mathbb{R}^N}{\operatorname{argmin}} \sum_{j=1}^{N} \left\| \boldsymbol{R}_{j,I} \circ \left(\boldsymbol{E}_j - d_j \underline{g}^T\right) \right\|_2^2 \tag{24}$$

$$\underline{d} = \left\{ d_j = \underset{d_j \in \mathbb{R}}{\operatorname{argmin}} \left\| \boldsymbol{R}_{j,I} \circ \left(\boldsymbol{E}_j - d_j \underline{g}^T\right) \right\|_2^2, j \in \{1..N\} \right\} \tag{25}$$

$$\underline{d} = \left\{ d_j = \underset{d_j \in \mathbb{R}}{\operatorname{argmin}} \left\| \boldsymbol{R}_{j,I} \circ \boldsymbol{E}_j - d_j . \boldsymbol{R}_{j,I} \circ \underline{g}^T \right\|_2^2, j \in \{1..N\} \right\} \tag{26}$$

$$\underline{d} = \left\{ d_j = \frac{\left\langle \boldsymbol{R}_{j,I} \circ \underline{g}^T, \boldsymbol{R}_{j,I} \circ \boldsymbol{E}_j \right\rangle}{\left\| \boldsymbol{R}_{j,I} \circ \underline{g}^T \right\|_2^2} \right\} \tag{27}$$

$$\underline{g} = \underset{\underline{g}\in\mathbb{R}^{|I|}}{\operatorname{argmin}} \left\| \boldsymbol{R}_I \circ \left( \boldsymbol{E} - \underline{d}\underline{g}^T \right) \right\|_2^2 \tag{28}$$

$$\underline{g} = \underset{\underline{g}\in\mathbb{R}^{|I|}}{\operatorname{argmin}} \left\| \boldsymbol{R}_I{}^T \circ \left( \boldsymbol{E}^T - \underline{g}\underline{d}^T \right) \right\|_2^2 \tag{29}$$

By matching eq. 29 with eq. 23, we can find a formula similar to eq. 27 for the coefficients.

# References


[PEA, 1998] (1998). Method for objective measurements of perceived audio quality.

[Abel and Smith, 1991] Abel, J. S. and Smith, J. O. (1991). Restoring a clipped signal. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Toronto, Canada.

[Adler et al., 2011] Adler, A., Emiya, V., Jafari, G., M., Elad, M., Gribonval, R., and Plumbley, M. D. (2011). Audio Inpainting. Rapport de recherche RR-7571, INRIA.

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

[Beck and Teboulle, 2009] Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202.

[Bertalmio et al., 2000] Bertalmio, M., Sapiro, G., Caselles, V., and Ballester, C. (2000). Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 417–424, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Blumensath and Davies, 2006] Blumensath, T. and Davies, M. (2006). Sparse and shift-invariant representations of music. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(1):50 – 57.

[Blumensath and Davies, 2008] Blumensath, T. and Davies, M. (2008). Gradient pursuits. *Signal Processing, IEEE Transactions on*, 56(6):2370 –2382.

[Dahimene et al., 2008] Dahimene, A., Noureddine, M., and Azrar, A. (2008). A simple algorithm for the restoration of clipped speech signal. *Informatica*, 32:183–188.

[Davis et al., 1997] Davis, G., Mallat, S., and Avellaneda, M. (1997). Adaptive greedy approximations. *Constructive Approximation*, 13:57–98. 10.1007/BF02678430.

[Donoho et al., 2006] Donoho, D. L., Tsaig, Y., Drori, I., and luc Starck, J. (2006). Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. Technical report.

[Engan et al., 1999] Engan, K., Aase, S. O., and Hakon Husoy, J. (1999). Method of optimal directions for frame design. In *Proceedings of the Acoustics, Speech, and Signal Processing, 1999. on 1999 IEEE International Conference - Volume 05*, ICASSP '99, pages 2443–2446, Washington, DC, USA. IEEE Computer Society.

[Etter, 1996] Etter, W. (1996). Restoration of a discrete-time signal segment by interpolation based on the left-sided and right-sided autoregressive parameters. *IEEE Transactions on Signal Processing*, 44(5):1124 –1135.

[Godsill and Rayner, 1998] Godsill, S. J. and Rayner, P. J. W. (1998). *Digital Audio Restoration - A Statistical Model-based Approach*. Springer-Verlag.

[Godsill et al., 2001] Godsill, S. J., Wolfe, P. J., and Fong, W. N. W. (2001). Statistical model-based approaches to audio restoration and analysis. *Journal of New Music Research*, 30(4).

[Goto et al., 2002] Goto, M., Hashiguchi, H., Nishimura, T., and Oka, R. (2002). Rwc music database: Popular, classical, and jazz music databases. In *In Proc. 3rd International Conference on Music Information Retrieval*, pages 287–288.

[Janssen et al., 1986] Janssen, A., Veldhuis, R., and Vries, L. (1986). Adaptive interpolation of discrete-time signals that can be modeled as autoregressive processes. *IEEE Trans. Acoustics, Speech and Sig. Proc.*, 34(2):317 – 330.

[Krstulovic and Gribonval, 2006] Krstulovic, S. and Gribonval, R. (2006). MPTK: Matching Pursuit made tractable. In *Proc. Int. Conf. Acoust. Speech Signal Process. (ICASSP'06)*, volume 3, pages III–496 – III–499, Toulouse, France.

[Lange et al., 2000] Lange, K., Hunter, D. R., and Yang, I. (2000). Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1):pp. 1–20.

[Mailhé et al., 2009] Mailhé, B., Gribonval, R., Bimbot, F., and Vandergheynst, P. (2009). Local orthogonal greedy pursuits for scalable sparse approximation of large signals with shift-invariant dictionaries. In Gribonval, R., editor, *SPARS'09 - Signal Processing with Adaptive Sparse Structured Representations*, Saint Malo, France. Inria Rennes - Bretagne Atlantique.

[Mailhé et al., 2008] Mailhé, B., Lesage, S., Gribonval, R., Bimbot, F., and Vandergheynst, P. (2008). Shift-invariant dictionary learning for sparse representations: extending K-SVD. In *16th EUropean SIgnal Processing COnference (EUSIPCO'08)*, page 5 p., Lausanne, Suisse.

[Mallat, 2009] Mallat, S. (2009). *A wavelet tour of signal processing: the sparse way*. Elsevier /Academic Press.

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

[Ofir et al., 2007] Ofir, H., Malah, D., and Cohen, I. (2007). Audio Packet Loss Concealment in a Combined MDCT-MDST Domain. *Signal Processing Letters, IEEE*, 14(12):1032 –1035.

[Pati et al., 1993] Pati, Y., Rezaiifar, R., and Krishnaprasad, P. (1993). Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40 –44 vol.1.

[Rubinstein et al., 2008a] Rubinstein, R., Rubinstein, R., Zibulevsky, M., and Elad, M. (2008a). Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. Technical report, Technion.

[Rubinstein et al., 2008b] Rubinstein, R., Zibulevsky, M., and Elad, M. (2008b). Learning sparse dictionaries for sparse signal representation. In *IEEE Transactions on Signal Processing. submitted. CHAPTER 1. SPARSE COMPONENT ANALYSIS*.

[Vincent et al., 2009] Vincent, E., Araki, S., and Bofill, P. (2009). The 2008 signal separation evaluation campaign: A community-based approach to large-scale evaluation. In *ICA*, Paraty, Brazil. Springer.

[Zeyde et al., 2010] Zeyde, R., Elad, M., and Protter, M. (2010). On single image scale-up using sparse-representations. In Boissonnat, J.-D., Chenin, P., Cohen, A., Gout, C., Lyche, T., Mazure, M.-L., and Schumaker, L. L., editors, *Curves and Surfaces*, volume 6920 of *Lecture Notes in Computer Science*, pages 711–730. Springer.

# Abstract

Recordings of audio often show undesirable alterations, mostly the presence of noise or the corruption of short parts. *Clipping*, or saturation, is one of such alterations. Several techniques have been developed in order to attempt the reversal of this corruption, achieving good but perfectible results. One of these techniques, developed in the METISS project-team, involves the use of *sparse representations*, a popular model in signal processing. The principle of sparse representations is to describe a high-dimensional data vector as a linear combination of a few prototype vectors, called *atoms,* selected from a large corpus called the *dictionary*. Building upon this technique, the aim of this internship is to define if and how can machine learning be applied on the dictionary in order to further enhance the results: what to learn on, with what learning algorithm, and with what kind of signals does it work?



**Keywords: Sparse representations, Declipping, Dictionary learning**