A survey and an extensive evaluation of popular audio declipping methods

Pavel Záviška, Pavel Rajmic, Alexey Ozerov, and Lucas Rencker

Abstract—**Dynamic range limitations in signal processing often lead to clipping, or saturation, in signals. The task of audio declipping is estimating the original audio signal, given its clipped measurements, and has attracted much interest in recent years. Audio declipping algorithms often make assumptions about the underlying signal, such as sparsity or low-rankness, and about the measurement system. In this paper, we provide an extensive review of audio declipping algorithms proposed in the literature. For each algorithm, we present assumptions that are made about the audio signal, the modeling domain, and the optimization algorithm. Furthermore, we provide an extensive numerical evaluation of popular declipping algorithms, on real audio data. We evaluate each algorithm in terms of the Signal-to-Distortion Ratio, and also using perceptual metrics of sound quality. The article is accompanied by a repository containing the evaluated methods.**

Index Terms—**audio clipping, saturation, declipping, model, sparsity, learning, optimization, evaluation, survey**

I. INTRODUCTION

CLIPPING is a non-linear signal distortion usually appearing when a signal exceeds its allowed dynamic range. As a typical instance, an analog signal that is digitized can be clipped in value when its original peak values go beyond the largest (or lowest) digit representation. For this reason, the effect is also called saturation.

Clipping in audio signals has a great negative effect on the perceptual quality of audio [1], and it reduces the accuracy of automatic speech recognition [2], [3] and other audio analysis applications. To improve the perceived quality of audio, a recovery of clipped samples can be made; this process is usually termed *declipping*.

Many audio declipping methods are available today. They are based on different modeling assumptions, tested on very different audio datasets, and evaluated by different methodologies. The goal of the article is to survey the existing approaches to audio declipping, categorize them and emphasize some interconnections. A no less important goal of this contribution is the numerical evaluation of selected audio declipping methods on a representative dataset and the provision of a freely available MATLAB toolbox. The subset of methods under consideration was selected based on our intention to cover different reconstruction techniques, on the popularity

P. Záviška and P. Rajmic are with the Signal Processing Laboratory, Brno University of Technology, Czech Republic. E-mail: rajmic@vutbr.cz.

A. Ozerov is with InterDigital, Cesson-Sévigné, France. E-mail: alexey.ozerov@interdigital.com.

L. Rencker is with the Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, UK. E-mail: l.rencker@surrey.ac.uk.

of the methods, and on the availability—or reproducibility—of their implementation (which go hand in hand in many cases). It is worth saying that the reconstruction quality is the primary focus in the evaluation, but comments on the speed of computation are provided as well.

In the case of the hard clipping, which is the degradation considered in this survey, the input signal exceeding the prescribed dynamic range $[-\theta_c, \theta_c]$ is limited in amplitude such that

$$y_n = \begin{cases} x_n & \text{for} \quad |x_n| < \theta_c, \\ \theta_c \cdot \operatorname{sgn}(x_n) & \text{for} \quad |x_n| \geq \theta_c, \end{cases} \quad (1)$$

where $[x_1, \ldots, x_N] = \mathbf{x} \in \mathbb{R}^N$ denotes the original (clean) signal and $[y_1, \ldots, y_N] = \mathbf{y} \in \mathbb{R}^N$ the observed clipped signal. The limiting constant θ_c is referred to as the clipping threshold (this article supposes that clipping is symmetric, without affecting generality). See Fig. 1 for an example of a clipped signal (and its various reconstructions).

Drawing on the terminology used in audio source separation [4] and machine learning in general [5], the audio declipping methods can be cast in two main categories:

- *unsupervised*, or *blind*, where the signal is recovered assuming some generic regularization (or modeling assumption) on what a natural audio signal should be like, but with no additional clean audio signals being involved,
- *supervised*, where signal recovery model parameters, or a part of them, are trained on (estimated from) clean audio examples, which should be similar to the audio sample to be recovered (e.g., all signals are speech signals).

To date, the vast majority of state-of-the-art audio declipping approaches are unsupervised, and thus, we limit this study to those approaches. However, supervised approaches are emerging as well, and they are mostly deep neural networks (DNNs)-based, trained to declip speech signals [6], [7], [8]. Supervised approaches are more specialized (since they are usually trained on particular classes of audio), and *potentially* more powerful, simply because more relevant information could be contained in the training set. As such, we believe that supervised learning is one of the potential and promising directions of evolution of research on audio declipping.

As for the unsupervised approaches, they often follow a generic path:

1) A *modeling domain* (e.g., time, analysis or synthesis, see Sec. II-A) is chosen.
2) A generic *model* regularizing an audio signal is specified in the chosen domain (e.g., autoregressive model, sparsity, group sparsity or low-rank structure).

3) *Model parameters* to be estimated from the clipped signal are specified (e.g., decomposition coefficients, coefficients and the dictionary, non-negative matrix factorization parameters, etc.).

4) A *criterion* linking the model parameters and observations to be optimized is specified (though the criterion is related to modeling, different choices are often possible; for instance, sparsity-assisted methods may penalize coefficients with the ℓ_0 or ℓ_1 norm). The criterion may or may not include the following ingredients:

   • *Clipped part consistency*: whether the clipping constraint in the missing part holds (see Sec. III).
   • *Reliable part consistency*: whether the reconstructed signal in the reliable part equals the observed clipped signal (see Sec. III).

5) A suitable *algorithm* to optimize the model criterion is chosen / constructed (e.g., orthogonal matching pursuit, expectation maximization, etc.).

6) Once the algorithm has terminated (typically, a fixed number of iterations are performed or a condition designed to check convergence is satisfied), the final signal is formed.

Most of state-of-the-art unsupervised audio declipping approaches characterized by the above-mentioned ingredients, including the approaches evaluated in this paper, are summarized in Table I.

In the case of multichannel audio (e.g., stereo) declipping may exploit correlations between different audio sources / objects in different channels, and this can improve the result over a straightforward, dummy solution of applying a single-channel declipping to each channel independently. This was for the first time investigated in [9], and then studied in [10] as well, though with a different approach. These works have shown that using the inter-channel dependencies can indeed improve the declipping performance. We do not evaluate those methods in this article, though, since there are only a few of them so far and such a task would require the creation of a particular multichannel dataset.

Roadmap of the article. Section II formulates the problem and prepares the notation used in further parts. A survey of declipping methods is given in Section III, while the methods selected for a thorough evaluation are described in more detail in Section IV. Then, the experiment and evaluation setup are explained in Section V, together with the results and their discussion. Finally, a conclusion is given and some perspectives for future research are indicated.

II. PROBLEM FORMULATION

In agreement with the clipping model (1), it is possible to divide the signal samples into three disjoint sets of indexes R, H, L such that $R \cup H \cup L = \{1, \ldots, N\}$ and, correspondingly, to distinguish the *reliable* samples (not influenced by clipping), samples *clipped from above* to the high clipping threshold θ_c and samples *clipped from below* to the low clipping threshold $(-\theta_c)$, respectively. The respective projection operators M_R, M_H and M_L (masks) are used to select samples
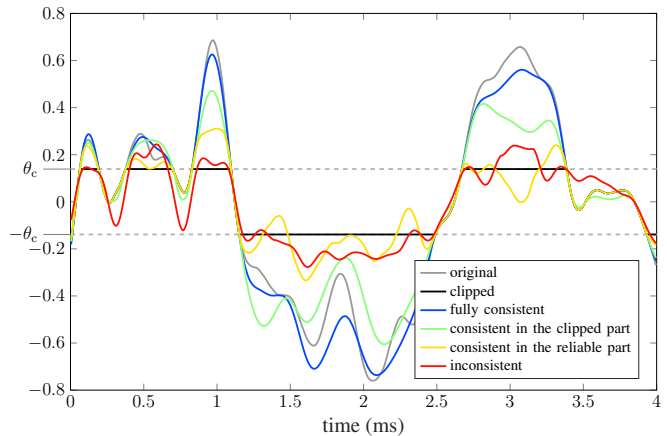

Fig. 1. Example of a clipped signal. Various recovery possibilities are depicted, showing different types of solution consistency (discussed in Sec. III).

from the corresponding set. With the mask operators, the following feasible sets can be defined:

$$\Gamma_R = \{\tilde{\mathbf{x}} \mid M_R\tilde{\mathbf{x}} = M_R\mathbf{y}\}, \tag{2a}$$
$$\Gamma_H = \{\tilde{\mathbf{x}} \mid M_H\tilde{\mathbf{x}} \geq \theta_c\}, \quad \Gamma_L = \{\tilde{\mathbf{x}} \mid M_L\tilde{\mathbf{x}} \leq -\theta_c\}, \tag{2b}$$
$$\Gamma = \Gamma_R \cap \Gamma_H \cap \Gamma_L, \tag{2c}$$

where we use the tilde notation $\tilde{\mathbf{x}}$ to represent arbitrary time-domain signals, while avoiding the confusion with $\mathbf{x}$, the clean signal. Note that these sets depend on the observation $\mathbf{y}$, since the masks do so too, hence formally we should write $\Gamma(\mathbf{y})$, for example, but we omit the dependence on the signal at most places for brevity.

The original dynamic range of the signal before clipping is typically unknown. However, if it is known or can at least be estimated, additional constraints like $M_H\tilde{\mathbf{x}} \leq \theta_{\max}$ and $M_L\tilde{\mathbf{x}} \geq -\theta_{\max}$ can be appended to (2) to further restrict the feasible set Γ. The scalars $\theta_{\min}$ and $\theta_{\max}$ represent the lower and upper bounds for the value of the signal. For example, [15] reports an improvement in signal recovery after such a trick for heavily clipped signals.

The declipping task is clearly ill-posed, since there is an infinite number of solutions that satisfy (2). Therefore, considering some additional information about the signal is crucial. That is where a signal model or statistical model comes into play, which regularizes the inverse problem.

A. Preliminaries and some notation

Most of the declipping methods rely on signal processing in a transformed domain. In such a context, $A: \mathbb{R}^N \to \mathbb{C}^P$ will denote the analysis operator, and $D: \mathbb{C}^P \to \mathbb{R}^N$ the synthesis operator. The operators are linear, it holds $P \geq N$, and the operators are connected through the relation $D = A^*$. The asterisk * denotes the adjoint operator. As an example, the common Discrete Fourier Transform (DFT) can play the role of A, while D is the inverse DFT. The DFT belongs to the class of unitary transfroms, where it holds $D = A^{-1}$.

Time-frequency audio processing, however, benefits from cases where $P > N$. For computational reasons, authors often restrict themselves to transforms referred to as the Parseval

TABLE I
CATEGORIZATION OF EXISTING SINGLE-CHANNEL UNSUPERVISED DECLIPPING APPROACHES. METHODS TREATED IN DETAIL AND EVALUATED ARE HIGHLIGHTED IN BOLD. TABLE ENTRIES THAT DID NOT FIT INTO A CLEAR CATEGORY ARE LEFT WITH THE "N/A" MARK.

Method	Modeling domain	Modeling assumptions	Model parameters	Optimization criterion	Clipping consistency	Rel. part consistency	Optimization algorithm
Janssen'86 [11]	AR	AR model	AR params.	ML	no	yes	EM
Abel'91 [12]	spectrum	limited bandwidth	band limit	several	yes	yes	N/A
Fong'01 [13]	N/A	AR model	N/A	AR coefs & correlation coefs	N/A	N/A	Monte Carlo
Dahimene'08 [14]	time	AR model	AR params.	least squares	no	yes	pseudoinverse
Adler'11 [15]	synthesis	sparsity	transform coefs	ℓ_0-min	yes	no	OMP
Weinstein'11 [16]	sparsity	sparsity	transform coefs	reweighted ℓ_1-min	yes	yes	CVX
Miura'11 [17]	synthesis	sparsity	transform coefs	ℓ_0-min	no	N/A	RVP (MP)
Kitić'13 [18]	synthesis	sparsity	transform coefs	ℓ_0-min	approximate	approximate	IHT
Defraene'13 [19]	synthesis	sparsity & psychoacoust.	transform coefs	ℓ_1-min	yes	no	CVX
Selesnick'13 [20]	time	smoothness	signal samples	regularized LS	no	no	explicit formula
Siedenburg'14 [21]	synthesis	social sparsity	transform coefs	social shrinkage	approximate	approximate	(F)ISTA
Kitić'14 [22]	analysis	sparsity	transform coefs	ℓ_0-min	yes	yes	ADMM
Jonscher'14 [23]	synthesis	sparsity	transform coefs	N/A	no	N/A	N/A
Bilen'15 [24]	analysis	low-rank NMF	NMF params.	ML	yes	yes	EM
Kitić'15 [25]	analysis & synthesis	sparsity	transform coefs	ℓ_0-min	yes	yes	ADMM
Harvilla'15 [26]	time	smoothness	signal samples	regularized LS	no	no	explicit formula
Takahashi'15 [27]	N/A	low rank	signal samples	quadratic	yes	yes	custom
Elvander'17 [28]	synthesis	sparsity	transform coefs	atomic norm min	yes	yes	SD
Rencker'18 [29]	synthesis	sparsity & learned dict.	transform coefs & dictionary	ℓ_0-min	approximate	approximate	alternate GD
Chantas'18 [30]	synthesis	sparsity	transform coefs	KL divergence	no	no	variational Bayes
Gaultier'19 [31]	analysis & synthesis	sparsity	transform coefs	ℓ_0-min	yes	yes	ADMM
Záviška'19 [32]	synthesis	sparsity	transform coefs	ℓ_0-min	yes	yes	ADMM
Záviška'19b [33]	synthesis	sparsity & psychoacoust.	transform coefs	ℓ_1-min	yes	yes	DR

Abbreviations: ADMM: Alternating Direction Method of Multipliers, AR: Autoregressive, CV: Condat–Vũ algorithm, CVX: convex opt. toolbox [34], DR: Douglas–Rachford alg., EM: Expectation–Maximization, (F)ISTA: (Fast) Iterative Shrinkage Thresholding Alg., GD: Gradient Descent, IHT: Iterative Hard Thresholding, KL: Kullback–Liebler, LS: Least Squares, ML: Maximum Likelihood, NMF: Nonnegative Matrix Factorization, (O)MP: (Orthogonal) Matching Pursuit, RVP: Recursive Vector Projection, SD: Semidefinite programming

tight frames [35], i.e., operators for which $DD^* = A^*A = \text{Id}$. Here Id stands for the identity operator. Unitary operators are clearly special cases of Parseval tight frames.

In synthesis-based signal models, one seeks for coefficients $\mathbf{z} \in \mathbb{C}^P$ that follow some prescribed properties, both directly in $\mathbb{C}^P$ and after synthesizing the coefficients into the signal $D\mathbf{z} \in \mathbb{R}^N$. In the analysis models, one seeks for a signal $\mathbf{x} \in \mathbb{R}^N$ that satisfies some properties both in $\mathbb{R}^N$ and after the analysis of $\mathbf{x}$ into coefficients, $A\mathbf{x} \in \mathbb{C}^P$ [36].

In finite-dimensional spaces (which is our case), the operators D and A can be identified with matrices. The matrix D corresponding to the synthesis is often called *the dictionary*, since its columns are the basic blocks in building the signal via their linear combination [37].

Since the methods covered by this survey concern exclusively audio, it is natural that the majority of the methods use transforms that map the signal to the time-frequency (TF) plane (and vice versa), such as the short-time Fourier transform (STFT), often referred to as the discrete Gabor transform (DGT) [35], [38]. Methods based on such time-frequency transforms work with (possibly overlapping) blocks of the signal. Such signal chunks are usually created by means of

time-domain windowing; note that this is the reason why we will speak about signal *windows*, alternatively about signal *blocks*, but not about the time *frames* of the signal, in order to avoid confusion with the above-introduced concept of frames in vector spaces. The TF coefficients are treated in the form of the vector $\mathbf{z}$ from the mathematical point of view, but note that for the user it is often more convenient to form a matrix $[z_{ft}]$ from $\mathbf{z}$. Its rows represent frequencies and its columns correspond to time-shifts of the windows. Such an arrangement corresponds to how spectrograms of audiosignals are usually visualized. The methods in Sections IV-I and IV-J will need to explicitly refer to individual signal blocks. For this sake, an additional index t will be used, such that, for instance, $\mathbf{y}_t$ will denote the t-th block of the clipped signal; in analogy to this, appending t to the masking operators and to the feasible sets will refer to their restriction in time, such as $M_{\mathrm{R}t}$ or $\Gamma(\mathbf{y}_t) = \Gamma_t$.

Norms of vectors will be denoted by $\|\cdot\|$, usually appended with the lower index characterizing the particular type of the norm. Using no index corresponds to the case of the operator norm (i.e., the largest singular value of the operator).

Many methods are based on a concept known as 'sparsity',

popularized in the last two decades [37], [39]. Exploiting sparsity means that within feasible declipping solutions, signals $D\mathbf{z}$ with a low number of nonzero elements of $\mathbf{z}$ are prioritized (synthesis model) or signals $\mathbf{x}$ with a low number of nonzeros in $A\mathbf{x}$ are preferred (analysis model) [39]. Mathematically, the $\|\cdot\|_0$ pseudo-norm is used to count the nonzeros of a vector.

Several methods presented below utilize the convex optimization; typically a sum of convex functions has to be minimized. In line with the recent trend in convex optimization, numerical solutions of such problems will be found using the so-called *proximal splitting algorithms* [40], [41], [42]. These are iterative schemes, usually with only a few principal—but rather simple—computations in each iteration. Each such computational step is related to the respective function in the sum separately. We recall that the proximal operator of a convex function f is the mapping

$$\text{prox}_f(\mathbf{x}) = \arg\min_{\mathbf{z}} \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|_2^2 + f(\mathbf{z}). \tag{3}$$

The concept provides a generalization of the common projection operator [40].

B. Consistency of declipping and relation to inpainting

Whichever technique is employed, every declipping method can be assigned to one of several classes, based on what is called consistency. A *fully consistent* method seeks a solution that is a member of the intersection $\Gamma = \Gamma_R \cap \Gamma_H \cap \Gamma_L$, or in other words, the recovered signal should equal the original samples in the reliable part and, at the same time, it should lie beyond the clipping thresholds in the clipped part. A solution *consistent in the reliable part* belongs to Γ_R, while a solution *consistent in the clipped part* is a member of $\Gamma_H \cap \Gamma_L$. A *fully inconsistent method* does not require a strict membership of the solution in any of the sets $\Gamma_R, \Gamma_H, \Gamma_L$. Even in such a case, a method either reflects the sets Γ_H and Γ_L (and thus the thresholds θ_c and $-\theta_c$ are taken into account in declipping) or these sets are completely ignored. Actually, if a user (or a particular application) decides not to take Γ_H and Γ_L into consideration, the declipping problem boils down to the so-called *audio inpainting* problem. Inpainting treats clipped samples simply as *missing*, hence ignoring potentially useful information; indeed, some of the presented methods in this survey are of this kind. Audio inpainting itself is an area, see for example [50], [51], [52] and the references therein. In our declipping experiment below, we included Janssen's method [11] as the representative of such methods (being actually very successful in audio inpainting).

Fig. 1 shows examples of different types of solution consistency. Consistent methods reflect the observed clipped signal and the clipping model, but the solutions are usually quite slow to compute. Inconsistency usually means a gain in speed in exchange for only approximate solutions (which might still be great for the human auditory system).

III.
DECLIPPING METHODS NOT SELECTED FOR EVALUATION

Apart from the detailed treatment of the methods selected for further numerical evaluation, this section is devoted to surveying the other declipping methods in the literature. We remind the reader that the characterization of all declipping methods is summarized in Table I.

Abel and Smith [12] discuss declipping of signals whose spectral band is limited (more than at the Nyquist frequency). This assumption is the key ingredient leading to a convex optimization program. The recovery uses oversampling and interpolation with sinc functions. The method is fully consistent, apart from the "noisy" variant treated at the end of [12].

Fong and Godsill [13] approach the declipping problem from the viewpoint of Bayesian statistical signal processing. The main assumption is the autoregressive nature of the signal, and to find the declipped samples, Monte Carlo particle filtering is utilized. The experiment follows a very simplified scenario (a single test on a very short speech sample).

Dahimene *et al.* [14] also start from the autoregressive (AR) assumption imposed on the signal. The paper forms a system of linear equations which is row-wise pruned in correspondence to the positions of the clipped samples. Two means of signal estimation are suggested: one based on ordinary least squares and the other based on Kalman filtering. This modeling does not guarantee any consistency in the clipped part.

The method introduced by Miura *et al.* [17] is based on a procedure coined recursive vector projection (RVP) by the authors. It turns out that RVP is actually the classical matching pursuit algorithm [43] restricted to reliable samples of the signal. Thus, it is a synthesis approach, with the dictionary described as the (possibly overcomplete) DFT. Since the clipping constraints are not taken into consideration, [17] is a method inconsistent in the clipped part, and its idea is actually quite similar to filling the missing samples of audio using the orthogonal matching pursuit in [44].

Jonscher *et al.* [23] introduce the Frequency Selective Extrapolation method. The signal is processed block by block. The clipped samples are treated as missing and their reconstruction is performed sequentially. The model behind the method can be understood as synthesis sparsity-based, with an overcomplete Fourier dictionary. Tests were carried out on speech signals only.

The method proposed by Takahashi *et al.* [45], [27] starts from an interesting observation that when the Hankel matrix is formed from a signal that follows the autoregressive (AR) model, the rank of this matrix is identical to the order of the respective AR process. Therefore, the approach aims at estimating the unknown but clipped elements of the Hankel matrix, whose rank is being minimized at the same time. After a series of simplifications, the authors formulate a convex optimization problem. The reported results look promising but unfortunately no data or codes are available.

Harvilla and Stern [26] introduce the RBAR (Regularized Blind Amplitude Reconstruction) method, which is reported to run in real time. The declipping task is formulated as an extended Tikhonov-regularized least squares problem, where the main idea is to penalize large deviations in the second difference of the signal, and at the same time to penalize deviations from the clipping level in the clipped part. The

experiments were carried out on speech signals, no codes are available.

Selesnick [20, Sec. 8.6] uses a similar regularizer. He proposes to minimize the energy of the the third difference of the signal, encouraging the filled-in data to have the form of a parabola. The numerical program is a least squares problem and as such it has a closed-form solution. Selesnick's method is however presented as one of the examples of the utilization of least squares and the paper does not provide any comparison with other methods.

Chantas et al. [30] build their declipping algorithm upon the Bayesian inference. They use a synthesis model, where the discrete cosine transform (DCT) coefficients are modeled as following the Student's distribution, complying with the assumption of their sparsity. No utilization of clipping constraints is involved. In addition, no computer implementation is available, unfortunately.

Elvander et al. [28] introduce probably the first approach that adapts the grid-less sparse recovery framework to the declipping problem. In brief, grid-less means that the dictionary does no longer contain countably many columns. In the case of [28], the continuous range of frequencies is available as the building blocks for the signal. A drawback of such an approach is that the resulting minimization problem is a semidefinite program, which can be computationally expensive.

In his PhD thesis, Gaultier [31] extends the idea of earlier algorithms based on hard thresholding [18], [22], [25]. The author works with the idea similar to the one published in [21], and introduces coefficients neighborhoods such that the TF coefficients are not processed individually (as is commonly done) but group-wise. This is shown to be beneficial, especially for mild clipping. A comprehensive study based of such an approach is contained in a recent article [46].

This survey considers only hard clipping governed by Eq. (1) but let us shortly mention the existence of the soft clipping (and the corresponding declipping methods). The transfer function of the soft clipping does not break suddenly at the points $-\theta_c$ and θ_c as in the case of hard clipping. Rather a certain transition interval is present around these spots that makes the transfer function smoother, resulting in less spectral distortion of clipping. The recovery of signals that have been soft-clipped is treated in [47], [48], [49], to name but a few.

IV. DECLIPPING METHODS SELECTED FOR EVALUATION

This section explains the principles of the methods that have been selected for the evaluation procedure. Each method comes with the algorithm in pseudocode (software implementation is addressed later in Section V). Some of the existing methods based on the synthesis sparsity are quite easily adaptable to the respective analysis counterpart; we include such unpublished variants in several cases to cover a wider range of methods. The order of the methods in this Section is chosen such that the ℓ_0-based are covered first, then the ℓ_1-based (optionally including psychoacoustics) are presented, then methods that can adapt to the signal, and as the last one the simple Janssen method [11] serving as the "anchor".

A. Constrained Orthogonal Matching Pursuit (C-OMP)

The approach to audio declipping proposed by Adler et al. [15] follows the same idea as the article [44] devoted to inpainting. The Orthogonal Matching Pursuit (OMP) is a well-known greedy-type algorithm [53], [54] used here as the first part of the procedure that approximates sparse coefficients in the NP-hard problem

$$\arg \min_{\mathbf{z}} \|\mathbf{z}\|_0 \ \text{s.t.} \ \begin{cases} \|M_{\mathrm{R}}\mathbf{y} - M_{\mathrm{R}}D\mathbf{z}\|_2 \leq \epsilon, \\ D\mathbf{z} \in \Gamma_{\mathrm{H}} \cap \Gamma_{\mathrm{L}}. \end{cases} \quad (4)$$

It is clear that (4) is a synthesis-based signal model and that it is clip-consistent, but inconsistent in the reliable part. The authors of [15] use an overcomplete discrete cosine transform in the role of the synthesis operator D.

The signal is cut into overlapping windows first, and the OMP is applied in each window separately. In the course of the OMP iterations, an increasing number of significant coefficients with respect to D are picked in a greedy fashion. Once such a subset of coefficients fulfils $\|M_{\mathrm{R}}\mathbf{y} - M_{\mathrm{R}}D\mathbf{z}\|_2 \leq \epsilon$, where $\epsilon > 0$ is the user-set precision required for the reliable part, the OMP stops. Notice that doing this is effectively performing the audio inpainting using OMP, i.e., ignoring the second condition in (4). However, as the very last step of the estimation, the current solution is updated using convex optimization. This makes the approach very slow, since it requires an iterative algorithm. The authors of [15] rely in this step on the CVX toolbox [34] in which (the subset of) D is handled in the matrix form, which decelerates the computations even more. After the algorithm is finished, the coefficients are synthesized using D. Individual blocks of the declipped signal are then put together in the time domain, using the common overlap–add procedure. The algorithm for a single window is summarized in Alg. 1.

Algorithm 1: Constrained OMP declipping [15] (C-OMP)

Input: D, $\mathbf{y} \in \mathbb{R}^N$, R, H, L

Parameters: $\epsilon > 0$

1 Using OMP, find an approximate solution, $\hat{\mathbf{z}}$, to problem
$\quad \arg\min_{\mathbf{z}} \|\mathbf{z}\|_0 \ \text{s.t.} \ \|M_{\mathrm{R}}\mathbf{y} - M_{\mathrm{R}}D_{\mathbf{w}}\mathbf{z}\|_2 \leq \epsilon$

2 Fix the support $\Omega \subseteq \{1, \ldots, P\}$ of $\hat{\mathbf{z}}$

3 Solve the constrained convex program
$\hat{\mathbf{z}}_\Omega = \arg\min_{\mathbf{z}_\Omega} \|M_{\mathrm{R}}(\mathbf{y} - D_\Omega\mathbf{z}_\Omega)\|_2 \ \text{s.t.} \ D_\Omega\mathbf{z}_\Omega \in \Gamma_{\mathrm{H}} \cap \Gamma_{\mathrm{L}}$

4 **return** $D_\Omega\hat{\mathbf{z}}_\Omega$

Some remarks should be made here: First, consider D as a matrix for the moment; the OMP requires that the columns of D have the same energy, i.e., the same ℓ_2 norm—this kind of normalization guarantees a fair selection of coefficients, at least for oscillatory signals, such as audio. To preserve such a condition, the problem on line 1 of Alg. 1 needs to weight the columns of D, which arises from the fact that the column subvectors used for estimation, $M_{\mathrm{R}}D$, do not contain the same energy. We denote this weighted synthesis $D_{\mathbf{w}}$. For more details, see the original paper [15] or the discussion on different types of weighting in [55].

Second, Alg. 1 uses the notation D_Ω for the synthesis operator restricted just to its columns contained in the set Ω, and, by analogy, for the restricted vector of coefficients $\mathbf{z}_\Omega$. There is in fact no need to weight the columns of D_Ω for the purpose of solving the problem on line 3.

Third, notice that the condition $\|M_R\mathbf{y} - M_R D\mathbf{z}\|_2 \leq \epsilon$ is in general violated after the update at line 3 because there might be no solution to the convex program that would satisfy it. Furthermore, hand in hand with this, notice that while D is usually chosen as a frame in $\mathbb{R}^N$, the restriction D_Ω does not have to inherit this property anymore, and this fact naturally applies to $M_R D_\Omega$ as well. In turn, when OMP finds $\Omega \subset \{1, \ldots, P\}$, there is no guarantee of the existence of *any* solution to the convex program. We will return to this issue in the experimental part since it will serve as an explanation of the strange numerical behavior of the C-OMP in some rare cases.

B. A-SPADE

The A-SPADE (Analysis SPArse DEclipper) was introduced by Kitić *et al.* in [25]. It is a natural successor to similar sparsity-based approaches [18] and [22], which are outperformed by A-SPADE [25].

The A-SPADE algorithm belongs to the ℓ_0-based family and it approximates the solution of the following NP-hard regularized inverse problem

$$\min_{\mathbf{x},\mathbf{z}} \|\mathbf{z}\|_0 \text{ s.t. } \mathbf{x} \in \Gamma(\mathbf{y}) \text{ and } \|A\mathbf{x} - \mathbf{z}\|_2 \leq \epsilon. \quad (5)$$

Here, $\mathbf{x} \in \mathbb{R}^N$ stands for the unknown signal in the time domain, and $\mathbf{z} \in \mathbb{C}^P$ contains the (also unknown) coefficients.

Parseval tight frames [35], i.e., $DD^* = A^*A = \text{Id}$, are considered. The processing of the signal is sequential, window by window. Due to the overlaps of windows, it suffices to use a simple TF transform like the DFT or the DCT (both possibly oversampled) which are Parseval tight frames.

The optimal solution to (5) in each window is approximated by means of the alternating direction method of mutipliers (ADMM). The resulting algorithm is in Alg. 2; for a detailed derivation and discussion, see [56]. The computational cost of the A-SPADE is dominated by the signal transformations; the algorithm requires one synthesis and one analysis per iteration. The hard thresholding $\mathcal{H}$ is a trivial operation. The projection on line 3 seeks for the signal $\mathbf{x} \in \Gamma$ whose analysis $A\mathbf{x}$ is the closest to $\bar{\mathbf{z}}^{(i+1)} - \mathbf{u}^{(i)}$. For tight frames, this task can be translated to an elementwise mapping in the time domain [56],

$$\left(\text{proj}_\Gamma(\mathbf{u})\right)_n = \begin{cases} y_n & \text{for } n \in R, \\ \max(\theta_c, u_n) & \text{for } n \in H, \\ \min(-\theta_c, u_n) & \text{for } n \in L, \end{cases} \quad (6)$$

$(\mathbf{u})_n$ denoting the nth element of the vector, i.e., $(\mathbf{u})_n = u_n$.

Compared with most available algorithms, it is fairly easy to tune the parameters. The variable k directly represents the number of selected coefficients in the hard-thresholding step. This number is growing in the course of iterations, driven by the parameters s and r (every r-th iteration, k is increased by s). The algorithm works really well with the basic setting, where k increases by one (i.e., $s = r = 1$).

Algorithm 2: A-SPADE algorithm [25]

Input: A, $\mathbf{y} \in \mathbb{R}^N$, R, H, L, $\epsilon > 0$
Parameters: $s, r \in \mathbb{N}$
Initialization: $\hat{\mathbf{x}}^{(0)} \in \mathbb{R}^N$, $\mathbf{u}^{(0)} \in \mathbb{C}^P$, $k = s$

1 **for** $i = 0, 1, \ldots$ **until** $\|A\mathbf{x} - \mathbf{z}\|_2 \leq \epsilon$ **do**
2    $\bar{\mathbf{z}}^{(i+1)} = \mathcal{H}_k\left(A\hat{\mathbf{x}}^{(i)} + \mathbf{u}^{(i)}\right)$
3    $\hat{\mathbf{x}}^{(i+1)} = \arg\min_{\mathbf{x}} \|A\mathbf{x} - \bar{\mathbf{z}}^{(i+1)} + \mathbf{u}^{(i)}\|_2^2$ s.t. $\mathbf{x} \in \Gamma$
4    $\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + A\hat{\mathbf{x}}^{(i+1)} - \bar{\mathbf{z}}^{(i+1)}$
5   **if** $(i+1) \bmod r = 0$ **then** $k = k + s$
6 **return** $\hat{\mathbf{x}}^{(i+1)}$

C. S-SPADE

Similarly to A-SPADE, Kitić *et al.* [25] also introduce a synthesis-based formulation,

$$\min_{\mathbf{x},\mathbf{z}} \|\mathbf{z}\|_0 \text{ s.t. } \mathbf{x} \in \Gamma(\mathbf{y}) \text{ and } \|\mathbf{x} - D\mathbf{z}\|_2 \leq \epsilon. \quad (7)$$

However, the S-SPADE algorithm from [25] has been shown in [32], [56] as actually solving a different optimization problem than (7). The same publications suggested a new version of the S-SPADE as the true counterpart of the A-SPADE, and showed its superiority in performance. Such an algorithm is in Alg. 3.

The computational complexity of Algs. 2 and 3 is the same: one synthesis, one analysis, the hard thresholding and an elementwise mapping per iteration is employed.

Algorithm 3: S-SPADE algorithm according to [32]

Input: D, $\mathbf{y} \in \mathbb{R}^N$, R, H, L, $\epsilon > 0$
Parameters: $s, r \in \mathbb{N}$
Initialization: $\hat{\mathbf{x}}^{(0)} \in \mathbb{R}^N$, $\mathbf{u}^{(0)} \in \mathbb{R}^N$, $k = s$

1 **for** $i = 0, 1, \ldots$ **until** $\|\mathbf{x} - D\mathbf{z}\|_2 \leq \epsilon$ **do**
2    $\bar{\mathbf{z}}^{(i+1)} = \mathcal{H}_k\left(D^*(\hat{\mathbf{x}}^{(i)} - \mathbf{u}^{(i)})\right)$
3    $\hat{\mathbf{x}}^{(i+1)} = \arg\min_{\mathbf{x}} \|D\bar{\mathbf{z}}^{(i+1)} - \mathbf{x} + \mathbf{u}^{(i)}\|_2^2$ s.t. $\mathbf{x} \in \Gamma$
4    $\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + D\bar{\mathbf{z}}^{(i+1)} - \hat{\mathbf{x}}^{(i+1)}$
5   **if** $(i+1) \bmod r = 0$ **then** $k = k + s$
6 **return** $\hat{\mathbf{x}}^{(i+1)}$

D. Declipping using weighted ℓ_1 minimization

The above methods are based on the ℓ_0 approach. Now we present several methods that rely on what is known as convex relaxation: their idea is to substitute the non-convex ℓ_0 pseudonorm with the "closest" convex norm, which is the ℓ_1 [37], [57]. The two declipping formulations in this section are quite basic, but to our knowledge they are treated only in [59] and [33]. In this article, they are included in the evaluation in their simple form but they also serve as the building block for algorithms from further sections.

Let us start with the simple synthesis-based task

$$\arg\min_{\mathbf{z}} \|\mathbf{w} \odot \mathbf{z}\|_1 \text{ s.t. } D\mathbf{z} \in \Gamma, \quad (8)$$

where D is the synthesis operator and $\odot$ denotes the elementwise product of two vectors. The vector $\mathbf{w} > 0$ is the vector of weights that can be set to all ones when no coefficients should be prioritized, but the larger an element of $\mathbf{w}$ is, the more penalized the corresponding coefficient in $\mathbf{z}$ is in the optimization. Due to the strict requirement $D\mathbf{z} \in \Gamma$, the approach based on (8) is fully consistent.

To find an appropriate algorithm to solve (8), it is convenient to rewrite it in an unconstrained form:

$$\arg\min_{\mathbf{z}} \|\mathbf{w} \odot \mathbf{z}\|_1 + \iota_\Gamma(D\mathbf{z}), \qquad (9)$$

where the hard constraint from (8) has equivalently been replaced by the indicator function ι,

$$\iota_C(\mathbf{u}) = \begin{cases} 0 & \text{for } \mathbf{u} \in C, \\ +\infty & \text{for } \mathbf{u} \notin C. \end{cases} \qquad (10)$$

Next, the observation is used that $\iota_\Gamma(D\mathbf{z}) = (\iota_\Gamma \circ D)(\mathbf{z}) = \iota_{\Gamma^*}(\mathbf{z})$, with Γ^* being the set of coefficients consistent with the clipping model, i.e., $\Gamma^* = \{\tilde{\mathbf{z}} \mid D\tilde{\mathbf{z}} \in \Gamma\}$; see also the definitions (2). The Douglas–Rachford algorithm (DR) [58] is able to find the minimizer of a sum of two convex functions of our type. The algorithm is presented in Alg. 4.

Algorithm 4: Douglas–Rachford (DR) alg. solving (8) [59]

Input: D, $\mathbf{y} \in \mathbb{R}^N$, $\mathbf{w} \in \mathbb{R}^P$, R, H, L
Parameters: $\lambda = 1$, $\gamma > 0$
Initialization: $\mathbf{z}^{(0)} \in \mathbb{C}^P$

1 **for** $i = 0, 1, \ldots$ **until** convergence **do**
2 $\quad \tilde{\mathbf{z}}^{(i)} = \text{proj}_{\Gamma^*} \mathbf{z}^{(i)}$ % using (11)
3 $\quad \mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} + \lambda\left(\text{soft}_{\gamma\mathbf{w}}(2\tilde{\mathbf{z}}^{(i)} - \mathbf{z}^{(i)}) - \tilde{\mathbf{z}}^{(i)}\right)$
4 **return** $D\mathbf{z}^{(i+1)}$

The algorithm iterates over two principal steps: The first is the projection onto Γ^*, which corresponds to the proximal operator of ι_{Γ^*} (recall the definition of prox in (3)). This projection is nontrivial and an explicit formula exists only in the case when DD^* is diagonal. According to [59], for Parseval tight frames it holds

$$\text{proj}_{\Gamma^*}(\mathbf{z}) = \mathbf{z} - D^*\left(D\mathbf{z} - \text{proj}_\Gamma(D\mathbf{z})\right), \qquad (11)$$

where proj_Γ is a trivial, elementwise time-domain mapping. The second step involves the soft thresholding $\text{soft}_{\tau\mathbf{w}}$ with the vector of thresholds $\tau\mathbf{w}$, which coincides with the proximal operator of the weighted ℓ_1-norm. The soft thresholding is an elementwise mapping defined by

$$\text{soft}_{\tau\mathbf{w}}(\mathbf{z}) = \mathbf{z} \odot \max\left(1 - \tau\mathbf{w} \odot \frac{1}{|\mathbf{z}|}, 0\right). \qquad (12)$$

The analysis counterpart of (8) reads

$$\arg\min_{\mathbf{x}} \|\mathbf{w} \odot A\mathbf{x}\|_1 \text{ s.t. } \mathbf{x} \in \Gamma, \qquad (13)$$

where A is the analysis operator. Unfortunately, the presence of A inside the weighted ℓ_1 norm prevents using the DR algorithm as above. The Chambolle–Pock (CP) algorithm [60] is able to cope with problems including a general linear

operator. Its particular form for signal declipping is shown in Alg. 5. There, clip is the Fenchel–Rockafellar conjugate of

---

Algorithm 5: Chambolle–Pock (CP) algorithm solving (13)

Input: A, $\mathbf{y} \in \mathbb{R}^N$, $\mathbf{w} \in \mathbb{R}^P$, R, H, L
Parameters: ζ, $\sigma > 0$ and $\rho \in [0, 1]$
Initialization: $\mathbf{x}^{(0)} \in \mathbb{R}^N$, $\mathbf{v}^{(0)} \in \mathbb{C}^P$

1 **for** $i = 0, 1, \ldots$ **until** convergence **do**
2 $\quad \mathbf{v}^{(i+1)} = \text{clip}_{\mathbf{w}}(\mathbf{v}^{(i)} + \sigma A\bar{\mathbf{x}}^{(i)})$
3 $\quad \mathbf{x}^{(i+1)} = \text{proj}_\Gamma(\mathbf{x}^{(i)} - \zeta A^* \mathbf{v}^{(i+1)})$
4 $\quad \bar{\mathbf{x}}^{(i+1)} = \mathbf{x}^{(i+1)} + \rho(\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)})$
5 **return** $\bar{\mathbf{x}}^{(i+1)}$

---

the soft thresholding, defined as

$$\text{clip}_{\mathbf{w}}(\mathbf{x}) = (\text{Id} - \text{soft}_{\mathbf{w}})(\mathbf{x}). \qquad (14)$$

For $\rho = 1$, the CP algorithm converges if $\zeta\sigma\|A\|^2 < 1$.

Looking at Algorithms 4 and 5, one recognizes that both have identical cost per iteration. The dominant operations are the transformations A and D.

E. Declipping in Sparseland (Rℓ_1CC)

Weinstein and Wakin [16] present four approaches to declipping, all based on sparsity. The basic synthesis model (8) is actually covered by the article under the acronym BPCC (Basis Pursuit with Clipping Constraints). In this section, we review the most successful method of [16] with coefficient reweighting, referred to as Rℓ_1CC (Reweighted ℓ_1 with Clipping Constraints) by the authors. It is again a fully consistent approach.

Rℓ_1CC follows up on a well-known idea from the field of sparse recovery / compressed sensing: To enhance the sparsity of the solution, a standard iterative program is performed, but repeatedly. During the repetitions, the actual weights $\mathbf{w}$ are being adapted based on the current temporary solution [61] and since each weight is inversely proportional to the magnitude of the respective coefficients, large coefficients are penalized less and less during the course of runs. For tiny coefficients, the opposite is true, leading to a sharper final sparsity and, in effect, to a significantly better bias of the solution [62], [63]. The described effect, however, is not achieved automatically; in some applications, the improvement can be large compared with the non-adaptive case [64], but sometimes it does not improve much [65] or even fails. It is worth noticing that it is not correct to say that Rℓ_1CC solves (8), see the discussion in [61].

To be more specific, Rℓ_1CC starts by solving the problem (8) with weights set to $\mathbf{w} = \mathbf{1}$. Based on the solution, $\mathbf{w}$ is recomputed and (8) is solved again, and again, until a reasonable convergence criterion is fulfilled. The authors of [16], however, provide no algorithm to solve (8).[1] We know from Sec. IV-D that the DR algorithm can be used. In turn, Rℓ_1CC is presented in Alg. 6, with reweighting performed

---

[1] Codes from https://github.com/aweinstein/declipping rely on CVX [34].

in step 4. Note that in practice, the number of outer loop repetitions should be controlled in order to avoid drop in performance; see the discussion in the evaluation part.

Algorithm 6: $R\ell_1CC$ using the Douglas–Rachford alg.

Input: D, $\mathbf{y} \in \mathbb{R}^N$, R, H, L
Parameters: $\epsilon > 0$
Initialization: $\mathbf{z}^{(0)} \in \mathbb{C}^P$, $\mathbf{w}^{(0)} = \mathbf{1}$

1 **for** $i = 0, 1, \ldots$ **until** convergence **do**
2   Solve (8) using Alg. 4 with $\mathbf{w}^{(i)}$ % returns $\mathbf{z}^{(i+1)}$
3     $\mathbf{w}^{(i+1)} = \frac{1}{|\mathbf{z}^{(i+1)}| + \epsilon}$ % update weights elementwise
4 **return** $D\mathbf{z}^{(i+1)}$

For this survey, we found it interesting to also include the analysis variant that was not considered in [16]. The procedure is analogous to the one just presented: Problem (13) is solved repeatedly, now by the CP algorithm, and the weights $\mathbf{w}$ are adapted depending on the last available solution, similar to how it is done in Alg. 6. However, the difference is that the solution of the CP algorithm is a time-domain signal; recomputing the weights thus requires the application of an additional analysis operator.

F. Social Sparsity

Siedenburg *et al.* [21] utilize for audio declipping the concept of the so-called 'social sparsity' [66], [67]. The plain sparsity induced by the ℓ_1 norm as the regularizer, used in the above sections, resulted in the soft thresholding of each coefficient individually in the respective algorithms. The social sparsity approach is more general: it allows the shrinkage of a coefficient, based also on the values of other coefficients, typically the coefficients in a kind of "neighborhood".

The particular design of the neighborhood depends heavily on the task to solve. For declipping, i.e., reverting a time-domain damage, TF neighborhoods that correspond to spreading in the direction of time are beneficial, since they help to share and leak information in the time direction. Such neighborhoods promote persistence in time, since with clipping, it makes more sense to focus on harmonic structures in audio than on transients.

Mathematically, the problem to solve is

$$\min_{\mathbf{z}} \left\{ \frac{1}{2} \|M_R D\mathbf{z} - M_R \mathbf{y}\|_2^2 + \frac{1}{2} \|h(M_H D\mathbf{z} - M_H \theta_c \mathbf{1})\|_2^2 + \right.$$
$$\left. + \frac{1}{2} \|h(-M_L D\mathbf{z} - M_L \theta_c \mathbf{1})\|_2^2 + \lambda R(\mathbf{z}) \right\}. \quad (15)$$

It is a synthesis model and it allows inconsistency of the reliable part (see the first term). The terms with h penalize the distance of the solution $D\mathbf{z}$ from the feasible set Γ in the clipped part; function h, called hinge, acts elementwise such that for each element of its input,

$$h(u) = \begin{cases} u & \text{for } u < 0 \\ 0 & \text{otherwise.} \end{cases} \quad (16)$$

The bold symbol $\mathbf{1}$ represents the vector of ones which is as long as the signal. Since the use of h in (15) does not guarantee that $D\mathbf{z}$ stays above and below the clipping thresholds, the method [21] is also inconsistent in the clipped part.

The first three terms in (15) are clearly differentiable, even with the Lipschitz-continuous gradient. Therefore, (15) can be treated as a sum of two functions, the second of them, R, being possibly non-smooth. This observation makes it possible to use standard optimization algorithms such as ISTA or FISTA [68], [69], [40], as outlined in Alg. 7.

Algorithm 7: ISTA-type Social sparsity declipper [21]

Input: $\mathbf{y} \in \mathbb{R}^N$, $\lambda > 0$, R, H, L, D; the shrinkage operator $\mathcal{S}$ (L / WGL / EW / PEW)
Parameters: $\gamma \in \mathbb{R}$, $\delta = \|DD^*\|$
Initialization: $\hat{\mathbf{z}}^{(0)}$, $\mathbf{z}^{(0)} \in \mathbb{C}^P$

1 **for** $i = 0, 1, \ldots$ **until** convergence **do**
2    $\mathbf{g}_1 = D^* M_R^* (M_R D\mathbf{z}^{(i)} - M_R \mathbf{y})$ % gradients
3    $\mathbf{g}_2 = D^* M_H^* h(M_H D\mathbf{z}^{(i)} - M_H \theta_c \mathbf{1})$
4    $\mathbf{g}_3 = D^* M_L^* h(-M_L D\mathbf{z}^{(i)} - M_L \theta_c \mathbf{1})$
5    $\hat{\mathbf{z}}^{(i+1)} = \mathcal{S}_{\lambda/\delta} \left(\mathbf{z}^{(i)} - \frac{1}{\delta}(\mathbf{g}_1 + \mathbf{g}_2 + \mathbf{g}_3) \right)$ % step, shrink
6    $\mathbf{z}^{(i+1)} = \hat{\mathbf{z}}^{(i+1)} + \gamma(\hat{\mathbf{z}}^{(i+1)} - \hat{\mathbf{z}}^{(i)})$ % extrapolate
7 **return** $D\hat{\mathbf{z}}^{(i+1)}$

In step 5 of Alg. 7, the gradients are added. Looking at the structure of the particular gradients on lines 2–4 reveals that in practical implementation, a much more effective way of computing $\mathbf{g}_1 + \mathbf{g}_2 + \mathbf{g}_3$ is possible, containing a single application of D and D^*. Another important trick that is not included in Alg. 7 for clarity of presentation is the warm start/adaptive restart strategy [70]: the authors of [21] found out that the overall convergence is significantly accelerated if ISTA is first run for a large λ for a few hundred iterations, then λ is decreased and so on, until the target value of λ from (15) is reached.

In Alg. 7, the shrinkage operator $\mathcal{S}$ plays the role of the proximal operator of R. The regularizer R should promote the expected structure of the TF coefficients of the signal. Paper [21] suggests using four types of social shrinkage of the TF coefficients $\mathbf{z}$, indexed by t (in time) and f (in frequency):

- LASSO (L): $\mathcal{S}_\lambda(z_{ft}) = z_{ft} \cdot \max\left(1 - \frac{\lambda}{|z_{ft}|}, 0\right).$

- Windowed Group LASSO (WGL):
  $\mathcal{S}_\lambda(z_{ft}) = z_{ft} \cdot \max\left(1 - \frac{\lambda}{\|\mathcal{N}(z_{ft})\|}, 0\right),$
  where $\mathcal{N}(z_{ft})$ denotes a vector formed from coefficients in the neighborhood of TF position ft.

- Empirical Wiener (EW):
  $\mathcal{S}_\lambda(z_{ft}) = z_{ft} \cdot \max\left(1 - \frac{\lambda^2}{|z_{ft}|^2}, 0\right).$

- Persistent Empirical Wiener (PEW):
  $\mathcal{S}_\lambda(z_{ft}) = z_{ft} \cdot \max\left(1 - \frac{\lambda^2}{\|\mathcal{N}(z_{ft})\|^2}, 0\right).$

Simple LASSO shrinkage is identical to soft thresholding and corresponds to the proximal operator of $R = \|\cdot\|_1$. The Empirical Wiener, also known as the non-negative garrote [71],

is better than LASSO in terms of bias [63] but it still operates on coefficients individually. EW is the proximal operator of a function R that has no explicit form, see for instance [72, Sec. 3.2] for more details. In contrast to LASSO and EW, both WGL and PEW involve TF neighborhoods, such that the resulting value of the processed coefficient z_{ft} depends on the energy contained in $\mathcal{N}(z_{ft})$. Again, the difference is only the second power used by the PEW. It is interesting to note that the study [73] proves that neither WGL nor PEW are proximal operators of *any* penalty R; in other words, the respective shrinkages are purely heuristic. Hand in hand with this, there are guarantees of convergence of Alg. 7 for LASSO and the Empirical Wiener if the extrapolation parameter γ is set properly [21], [40], while setting it in the case of WGL and PEW requires trial tuning.

The experiments in [21] give evidence that out of the four choices, only PEW and EW are well-performing and they outperform both the OMP [15] and C-IHT [18] approaches.

G. Perceptual Compressed Sensing

The approach by Defraene *et al.* [19] was for a long time the only one to include psychoacoustics in declipping (both in the model itself and in the evaluation). Although the approach is not quite related to the compressed sensing, we refer to the method using the same name as the authors coined. The following optimization problem is solved:

$$\min_{\mathbf{z}} \left\{ \frac{1}{2} \|M_{\mathrm{R}} D\mathbf{z} - M_{\mathrm{R}}\mathbf{y}\|_2^2 + \lambda\|\mathbf{w} \odot \mathbf{z}\|_1 \right\} \text{ s.t. } D\mathbf{z} \in \Gamma_{\mathrm{H}} \cap \Gamma_{\mathrm{L}}. \tag{17}$$

This might look like just another variation of the synthesis-based declipping, but the main difference from the other methods is that the weights $\mathbf{w}$ are computed based on a human perception model. Note that with respect to our terminology, this method is *consistent in the clipped part*.

To be more specific about the method, the signal is processed window-by-window. Ignoring the introduction of correct notation, task (17) is solved independently for the signal chunks given by windowing. The recovered signal is obtained by the application of the synthesis D to the found coefficients, and by reusing the reliable samples at positions given by the set R. Once all the windows are processed this way, the final signal is obtained via the overlap–add procedure.

The psychoacoustic model enters the task through $\mathbf{w}$. The authors of [19] rely on the MPEG-1 Layer 1 psychoacoustic model 1 [74], [75], which computes the instantaneous masking curve based on the incoming signal and on the absolute threshold of hearing. In short, such a curve informs us about the spectral components in the signal that will not be perceived when other strong components are present. This effect is commonly known as the instantaneous masking or frequency masking. Inspired by this effect, $\mathbf{w}$ is set as the inverse of this curve (the curve in dB is non-negative, which justifies such an approach from the mathematical point of view). Application of such weights can be interpreted as discouraging the introduction of distinctively audible new spectral components that are not present in the original signal. On the other hand, the introduction of less audible or inaudible spectral components is tolerated to a greater extent [19].

Worth noticing is that a correctly treated masking curve: 1) should be computed based on the original signal, not available in practice, 2) should be applied to the very current window of the signal. The authors of [19] cope with both the issues in such a way that they recurrently use the just declipped window as the base for calculating the masking curve, which is then applied in declipping the subsequent window.

In terms of the numerical treatment of (17), [19] proposes an algorithm termed PCSL1. Its core, optimization part, refers to the CVX toolbox [34], but no particular codes for PCSL1 are available, unfortunately. After several unsuccessful trials with CVX, we decided to solve the problem with the so-called generic proximal algorithm introduced by Condat and Vũ [41], [76]. Such an algorithm, in the following abbreviated as the CV algorithm, is able to solve convex problems with more than two terms, possibly even containing linear operators. This is the case of (17), indeed. Alg. 8 presents the particular shape of the CV algorithm for declipping. The projection onto $\Gamma_{\mathrm{H}} \cap \Gamma_{\mathrm{L}}$ is done using the second and third lines of (6). Algorithm 8 is guaranteed to converge if $\sigma < \tau^{-1} - 1/2$.

Algorithm 8: Condat–Vũ (CV) algorithm solving (17)

Input: D, $\mathbf{y} \in \mathbb{R}^N$, $\mathbf{w} \in \mathbb{R}^P$, $\lambda > 0$, R, H, L

Parameters: $\sigma, \tau > 0$ and $\rho \in (0, 1]$

Initialization: $\mathbf{z}^{(0)} \in \mathbb{C}^P$, $\mathbf{u}^{(0)} \in \mathbb{R}^N$

1 **for** $i = 0, 1, \ldots$ **until** convergence **do**

2 $\quad \tilde{\mathbf{z}}^{(i+1)} =$
$\quad \mathrm{soft}_{\tau\lambda\mathbf{w}} \left(\mathbf{z}^{(i)} - \tau D^* \left[M_{\mathrm{R}}^* M_{\mathrm{R}} (D\mathbf{z}^{(i)} - \mathbf{y}) + \mathbf{u}^{(i)} \right] \right)$

3 $\quad \mathbf{z}^{(i+1)} = \rho\tilde{\mathbf{z}}^{(i+1)} + (1-\rho)\mathbf{z}^{(i)}$

4 $\quad \mathbf{p}^{(i+1)} = \mathbf{u}^{(i)} + \sigma D(2\tilde{\mathbf{z}}^{(i+1)} - \mathbf{z}^{(i)})$ % auxiliary

5 $\quad \tilde{\mathbf{u}}^{(i+1)} = \mathbf{p}^{(i+1)} - \sigma\,\mathrm{proj}_{\Gamma_{\mathrm{H}}\cap\Gamma_{\mathrm{L}}}\left(\mathbf{p}^{(i+1)}/\sigma\right)$

6 $\quad \mathbf{u}^{(i+1)} = \rho\tilde{\mathbf{u}}^{(i+1)} + (1-\rho)\mathbf{u}^{(i)}$

7 **return** $D\mathbf{z}^{(i+1)}$

H. Psychoacoustically motivated ℓ_1 minimization

The second method that involves psychoacoustics, by Záviška *et al.* [33], is similar to the above (Section IV-G), but it is designed as completely consistent. Recall that this means that the declipped signal should belong to the set Γ defined in (2). The problem solved in [33] is actually identical to (8), but the weights are now derived from the human perception model. It is a synthesis-based signal model, and again, the Douglas–Rachford algorithm presented in Alg. 4 can be used to find the numerical solution (with the efficient projection onto Γ^* in the case that D is a tight frame).

Unlike [19], the paper [33] discusses multiple ways of choosing the weights $\mathbf{w}$. Besides the basic inversion, there are several other options of "inverting" the masking curve that have been introduced and evaluated. Surprisingly, the best declipping results were obtained using weights which simply grow quadratically with frequency! Such an option is not

psychoacoustically inspired at all, but its success might be explained by the fact that signals after clipping have a very rich spectrum, while the spectra of original signals decay with increasing frequency. In other words, regularizing the spectrum in such a way (in addition to sparsity) seems to be more powerful than modeling delicate perceptual effects. For the experiments in Sec. V, just this "parabola" option was selected.

Motivated by such an interesting observation, we also employed these quadratic weights $\mathbf{w}$ in the method from Sec. IV-G. See more in the evaluation part of the article.

I. Dictionary Learning approach

In the next two sections, we consider T windows $\mathbf{y}_1, \ldots, \mathbf{y}_T$ of the clipped signal $\mathbf{y}$, and their corresponding consistency sets $\Gamma_t = \Gamma(\mathbf{y}_t)$.

Sparsity-based methods reviewed so far use fixed and known synthesis operators, such as the DCT or Gabor transforms. However, another approach consists in *adapting* D to the observed data. Learning the dictionary (we prefer the term *dictionary* since D will be treated in its matrix form from now on), rather than using an off-the-shelf one, has shown improvements in inverse problems such as denoising or in-painting [77], [39]. Dictionary learning (DL) from clipped observations has been formulated by Rencker *et al.* [78], [29]. Given a collection of T clipped signals $\mathbf{y}_1, \ldots, \mathbf{y}_T$ (typically corresponding to T overlapping time windows extracted from a signal), dictionary learning from clipped observations can be formulated as:

$$\min_{\mathbf{z}_t, D} \sum_{i=1}^{T} \mathrm{d}(D\mathbf{z}_t, \Gamma_t)^2 \text{ s.t. } \|\mathbf{z}_t\|_0 \leq K, \ t = 1, \ldots, T, \quad (18)$$

where $\mathrm{d}(D\mathbf{z}_t, \Gamma_t)$ is the Euclidean distance of $D\mathbf{z}_t$ to the set Γ_t, and Γ_t is the feasibility set corresponding to the signal $\mathbf{y}_t$ (as defined in (2)). Note that using the notation in Sec. II, $\mathrm{d}(\cdot, \Gamma_t)^2$ is equivalent to the data-fidelity term in (15), and is convex, differentiable with Lipschitz gradient thanks to the convexity of Γ_t. DL algorithms typically alternate between optimizing $\mathbf{z}_1, \ldots, \mathbf{z}_T$ with D fixed (sparse coding step), and optimizing D with $\mathbf{z}_1, \ldots, \mathbf{z}_T$ fixed (dictionary update step) [39].

The sparse coding step solves, for each t independently:

$$\min_{\mathbf{z}_t} \mathrm{d}(D\mathbf{z}_t, \Gamma_t)^2 \quad \text{s.t.} \quad \|\mathbf{z}_t\|_0 \leq K, \quad (19)$$

which can be aproximated using consistent Iterative Hard Thresholding (IHT). Consistent IHT, proposed in [18], is a simple algorithm that iterates:

$$\mathbf{z}_t \leftarrow \mathcal{H}_K(\mathbf{z}_t + \mu D^\top (D\mathbf{z} - \mathrm{proj}_{\Gamma_t}(D\mathbf{z}_t)), \quad (20)$$

which corresponds to a gradient descent step (with the parameter μ), followed by the hard thresholding. The ℓ_0 constraint in (18) can also be relaxed into an ℓ_1 constraint, in which case the sparse coding step would correspond to an ISTA-type algorithm in Alg. 7.

The dictionary update step is formulated as:

$$\min_{D \in \mathcal{D}} \sum_{t=1}^{T} \mathrm{d}(D\mathbf{z}_t, \Gamma_t)^2, \quad (21)$$

which can be solved using (accelerated) gradient descent. Note that D is constrained to belong to $\mathcal{D} = \{D = [\mathbf{d}_1, \ldots, \mathbf{d}_P] \mid \|\mathbf{d}_p\|_2 \leq 1 \text{ for } p = 1, \ldots, P\}$ in order to avoid scaling ambiguity.

The overall dictionary learning algorithm is presented in Algorithm 9. When $\mathbf{y}_1, \ldots, \mathbf{y}_T$ correspond to overlapping windows extracted from a given signal, each window can be recovered using the estimated dictionary and sparse coefficients as $\hat{D}\hat{\mathbf{z}}_1, \ldots, \hat{D}\hat{\mathbf{z}}_T$. The overall signal can then be estimated using overlap–add.

Algorithm 9: Dictionary learning algorithm for declipping [29]

Input: $\mathbf{y}_1, \ldots, \mathbf{y}_T \in \mathbb{R}^N, \ R, \ H, \ L$
Parameters: $K, \ P$
Initialization: $\mathbf{z}_1^{(0)}, \ldots, \mathbf{z}_T^{(0)} \in \mathbb{R}^P, \ D^{(0)} \in \mathbb{R}^{N \times P}$

1 **for** $i = 0, 1, \ldots$ **until** convergence **do**

2   Solve for $t = 1, \ldots, T$, using e.g., consistent IHT:
     $\mathbf{z}_t^{(i+1)} = \arg\min_{\mathbf{z}_t} \mathrm{d}(D^{(i)}\mathbf{z}_t, \Gamma_t)^2$ s.t. $\|\mathbf{z}_t\|_0 \leq K$

3   Solve using, e.g., accelerated gradient descent:
     $D^{(i+1)} = \arg\min_D \sum_{t=1}^{T} \mathrm{d}(D\mathbf{z}_t^{(i+1)}, \Gamma_t)^2$

4 **return** $D^{(i+1)}\mathbf{z}_1^{(i+1)}, \ldots, D^{(i+1)}\mathbf{z}_T^{(i+1)}$

J. Nonnegative Matrix Factorization

Another approach proposed recently by Bilen *et al.* [24], [79] is based on the nonnegative matrix factorization (NMF). This is a dictionary learning approach too, but instead of learning a dictionary of the waveform, it learns a nonnegative dictionary together with nonnegative decomposition coefficients to approximate the unknown power spectrogram of the original signal. This corresponds to the assumption that the power spectrogram is approximately low-rank. Given that the power spectrogram is a phase-free representation, this modeling is phase-invariant, thus allowing using a dictionary of a considerably smaller size than the dictionary size in the approach presented in Section IV-I.

NMF modeling is defined on the latent clean signal power spectrogram obtained from the analysis short-time Fourier transform (STFT) coefficients. Note that it is also possible to decompose power spectrograms of synthesis coefficients, as in [80], though this has not yet been done for audio declipping but for a related problem of compressed sensing recovery [80]. More specifically, the analysis NMF approach assumes that the power spectrogram nonnegative matrix $\mathbf{P} = [p_{ft}]_{f,t=1}^{F,T}$ (with $p_{ft} = |z_{ft}|^2$, and z_{ft} being clean signal STFT coefficients) is approximated as

$$\mathbf{P} \approx \mathbf{V} = \mathbf{W}\mathbf{H}, \quad (22)$$

with $\mathbf{W} \in \mathbb{R}_+^{F \times K}$ and $\mathbf{H} \in \mathbb{R}_+^{K \times T}$ nonnegative matrices, and $K > 0$ usually chosen much smaller than F and T, $K \ll \min(F, T)$. The matrix $\mathbf{W}$ can be understood as the power spectrum dictionary (columns of $\mathbf{W}$ being characteristic spectral patterns), while $\mathbf{H}$ contains the corresponding nonnegative decomposition (activation) coefficients.

Note that the modeling in (22) is not yet well defined, since the approximation is not specified mathematically and the power spectrogram $\mathbf{P}$ is unknown. To specify it properly, it is proposed in [24], [79] to resort to the maximum likelihood (ML) optimization under a probabilistic Gaussian formulation of Itakura Saito (IS) NMF [81]. To simplify the formulation here, let all signals be considered either in the time domain (windowed, with overlap) or in the frequency domain, while the two domains are related by the DFT for each time block separately. The DFT, denoted A here, $A\colon \mathbb{C}^F \to \mathbb{C}^F$, is unitary, and the number of frequency channels F is identical to the number of time-domain samples. Let $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_T]$ and $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ denote the windowed versions of the clipped and original (unknown) signals, respectively, and $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_T]$ the STFT of the original signal ($\mathbf{z}_t = A\mathbf{x}_t$). It is assumed that the coefficients in $\mathbf{Z}$ are all mutually independent, and each coefficient z_{ft} follows a complex circular zero-mean Gaussian distribution $z_{ft} \sim \mathcal{N}_{\mathrm{c}}(0, v_{ft})$ with $\mathbf{V} = [v_{ft}]_{f,t}$ being a low-rank power spectrogram approximation specified in (22). NMF model parameters are estimated optimizing the ML criterion (see [24] for details)

$$(\mathbf{W}, \mathbf{H}) = \arg \max_{\mathbf{W}', \mathbf{H}'} \; p(\mathbf{Y}|\mathbf{W}', \mathbf{H}') \qquad (23)$$

via the generalized expectation–maximization (GEM) algorithm [82] with multiplicative update (MU) rules [81]. The final windowed signal block estimate $\widehat{\mathbf{X}}$ is recovered via Wiener filtering, see (24).[2] This is altogether summarized in Alg. 10, where $M_{\mathrm{R}t}$ denotes the restriction of the operator M_{R} to block t, and all operators (e.g., $M_{\mathrm{R}t}$ or A^*), when applied to matrices, are applied column-wise. It should be highlighted that, though the NMF modeling (22) is defined on the signal power spectrogram, the signal is reconstructed with both amplitude and phase since the Wiener filtering (24) with a complex-valued Wiener gain (matrix $\Sigma^*_{M_{\mathrm{R}t}\mathbf{y}_t\mathbf{z}_t}\Sigma^{-1}_{M_{\mathrm{R}t}\mathbf{y}_t M_{\mathrm{R}t}\mathbf{y}_t}$) maps from the time domain to the complex-valued STFT domain.

Note that the consistency in the clipped part is not satisfied in Algorithm 10, and it is difficult to satisfy it properly since with this constraint the posterior distribution of $\mathbf{x}_t$ (given the observations and the NMF model) is not Gaussian any more. To take the constraint into account, an ad hoc strategy has been proposed in [24], [79]. This strategy consists in checking in step 4 whether the clipping constraint is satisfied. If not, the samples of those blocks $\hat{\mathbf{x}}_t$ for which it is not satisfied are projected on the corresponding clipping thresholds, dynamically added to the reliable set, and for those blocks the steps 3 and 4 are repeated again, and, if necessary, iterated till clipping constraint is completely satisfied in step 4. Whenever the iteration of the main algorithm starts over again, the reliable set is re-initialized (see [24], [79]).

K. Janssen's autoregressive interpolation

The Janssen method [11] published back in 1986 and thoroughly discussed recently in [83] relies on the autoregressive (AR) signal model. It assumes that a particular signal sample

---

[2] Estimating windowed signal blocks results in a problem relaxation [79] since the overlapping frames are clearly not independent, but those dependencies are not exploited during estimation.

Algorithm 10: NMF GEM algorithm [24]

Input: $\mathbf{y}_1, \ldots, \mathbf{y}_T \in \mathbb{R}^F,\ R,\ H,\ L$

Parameters: $K > 0$

Initialization: $\mathbf{W}^{(0)} \in \mathbb{R}_+^{F \times K},\ \mathbf{H}^{(0)} \in \mathbb{R}_+^{K \times T}$ (random)

1   $\mathbf{V}^{(0)} = \mathbf{W}^{(0)}\mathbf{H}^{(0)}$

2   **for** $i = 0, 1, \ldots$ **until** convergence **do**

3   Estimate posterior power spectrogram $\mathbf{P} = [p_{ft}]$:

     $\hat{p}_{ft} = |\hat{z}_{ft}^{(i+1)}|^2 + \widehat{\Sigma}_{\mathbf{z}_t\mathbf{z}_t}(f, f)$,

     where (f, f) picks the f-th diagonal matrix entry and

$$\hat{\mathbf{z}}_t^{(i+1)} = \Sigma^*_{M_{\mathrm{R}t}\mathbf{y}_t\mathbf{z}_t}\Sigma^{-1}_{M_{\mathrm{R}t}\mathbf{y}_t M_{\mathrm{R}t}\mathbf{y}_t}M_{\mathrm{R}t}\mathbf{y}_t, \qquad (24)$$

     $\widehat{\Sigma}_{\mathbf{z}_t\mathbf{z}_t} = \Sigma_{\mathbf{z}_t\mathbf{z}_t} - \Sigma^*_{M_{\mathrm{R}t}\mathbf{y}_t\mathbf{z}_t}\Sigma^{-1}_{M_{\mathrm{R}t}\mathbf{y}_t M_{\mathrm{R}t}\mathbf{y}_t}\Sigma_{M_{\mathrm{R}t}\mathbf{y}_t\mathbf{z}_t}$,

     with

     $\Sigma_{\mathbf{z}_t\mathbf{z}_t} = \mathrm{diag}\left([v_{ft}^{(i)}]_f\right),\ \Sigma_{M_{\mathrm{R}t}\mathbf{y}_t\mathbf{z}_t} = M_{\mathrm{R}t}A^*\Sigma_{\mathbf{z}_t\mathbf{z}_t}$,

     $\Sigma_{M_{\mathrm{R}t}\mathbf{y}_t M_{\mathrm{R}t}\mathbf{y}_t} = M_{\mathrm{R}t}A^*\Sigma^*_{M_{\mathrm{R}t}\mathbf{y}_t\mathbf{z}_t}$.

4   Compute: $\hat{\mathbf{x}}_1^{(i+1)} = A^*\hat{\mathbf{z}}_1^{(i+1)}, \ldots, \hat{\mathbf{x}}_T^{(i+1)} = A^*\hat{\mathbf{z}}_T^{(i+1)}$

5   Update NMF parameters using MU rules:

     $\mathbf{W}^{(i+1)} = \mathbf{W}^{(i)} \odot \dfrac{\left(\left[\mathbf{W}^{(i)}\mathbf{H}^{(i)}\right]^{\cdot -2}\odot\widehat{\mathbf{P}}\right)\left(\mathbf{H}^{(i)}\right)^\top}{\left[\mathbf{W}^{(i)}\mathbf{H}^{(i)}\right]^{\cdot -1}\left(\mathbf{H}^{(i)}\right)^\top}$,

     $\mathbf{H}^{(i+1)} = \mathbf{H}^{(i)} \odot \dfrac{\left(\mathbf{W}^{(i+1)}\right)^\top\left(\left[\mathbf{W}^{(i+1)}\mathbf{H}^{(i)}\right]^{\cdot -2}\odot\widehat{\mathbf{P}}\right)}{\left(\mathbf{W}^{(i+1)}\right)^\top\left[\mathbf{W}^{(i+1)}\mathbf{H}^{(i)}\right]^{\cdot -1}}$,

     with $\odot$ and $[\cdot]^{\cdot b}$ denoting element-wise matrix product and power, all divisions being element-wise as well, and $[\cdot]^\top$ denoting the transpose.

6   Update: $\mathbf{V}^{(i+1)} = \mathbf{W}^{(i+1)}\mathbf{H}^{(i+1)}$

7   **return** $\hat{\mathbf{x}}_1^{(i+1)}, \ldots, \hat{\mathbf{x}}_T^{(i+1)}$ after applying the corresponding synthesis window and overlap–add to get the signal in time domain.

---

can be induced via a fixed linear combination of preceding samples. The coefficients in such a combination are the AR coefficients and their total number is called the order of the AR model. The model can be alternatively interpreted such that the audio signal is generated by a Gaussian white noise filtered by an all-pole filter.

In practice, the AR model can be successfully applied to signals containing harmonic components. The Janssen method cannot handle the clipping constraints; hence in declipping, it is only possible to use it in order to replace the clipped samples by the values linearly estimated from the reliable samples. In that regard, the Janssen method actually belongs to the audio inpainting methods.

Despite its simplicity and age, the algorithm is a strong competitor of the most recent audio inpainting methods, [50]. That is why we decided to consider it within our evaluation.

V. EVALUATION

This section compares the selected audio declipping methods from the previous section in terms of the quality of reconstruction. First, the design of the experiments is described, along with the characterization of the audio dataset.

The evaluation metrics used to objectively assess the quality of reconstruction are presented then. Subsection V-C contains details about the algorithms from the practical viewpoint, such as settings of the parameters and comments on the behavior of the algorithms. Finally, the results are presented and discussed.

A. Experiment design and the dataset

The audio database used for the evaluation consists of 10 musical excerpts in mono, sampled at 44.1 kHz, with an approximate duration of 7 seconds. They were extracted from the EBU SQAM database.[3] The excerpts were thoroughly selected to cover a wide range of audio signal characteristics. Since a significant number of the methods are based on signal sparsity, the selection took care that different levels of sparsity were included in the signals (w.r.t. the Gabor transform). We have selected the sounds of the violin, clarinet, basson, harp, glockenspiel, celesta, accordion, guitar, piano, and the wind ensemble.

To the best of our knowledge, the only declipping experiments including audio sampled at 44.1 kHz were carried in [19] and [33], while the others used audio at 16 kHz at the most. This survey thus provides the very first large-scale experiment for a high-quality sampled audio.

The input data were clipped in agreement with the model in Eq. (1), using clipping levels that were chosen to lead to 7 different input signal-to-distortion ratios (SDR). The SDR for two signals $\mathbf{u}$ and $\mathbf{v}$ is defined as

$$\mathrm{SDR}(\mathbf{u}, \mathbf{v}) = 20 \log_{10} \frac{\|\mathbf{u}\|_2}{\|\mathbf{u} - \mathbf{v}\|_2}. \tag{25}$$

Recall that $\mathbf{x}$ denotes the original and $\mathbf{y}$ the clipped signal; hence, the input SDR is computed as $\mathrm{SDR}(\mathbf{x}, \mathbf{y})$.

With respect to the human perception of clipping severity, the SDR is more meaningful than treating signals according to the clipping levels or according to the percentage of clipped samples [31]. The particular input SDR levels are chosen to cover the range from very harsh clipping to mild but still noticeable clipping. The specific values along with the respective percentages of clipped samples are visualized in Fig. 2. Since the input SDR is used, there is no need to peak-normalize the audio samples before processing because the number of the clipped samples remains the same, independently of scaling.

All the data have been processed and evaluated using MATLAB in double precision, and therefore there is no additional distortion caused by quantization during the process.

B. Evaluation metrics

To evaluate the reconstruction quality, we use several metrics.

1) Signal-to-Distortion Ratio: Firstly, we utilize the signal-to-distortion ratio (SDR), which is one of the simplest, nonetheless, one of the most used methods. It expresses the physical quality of reconstruction, i.e., how (numerically) close to the ground truth $\mathbf{x}$ the recovered signal $\hat{\mathbf{x}}$ is.

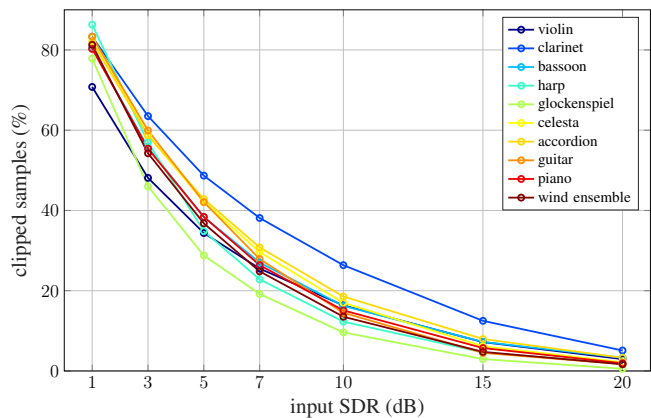[3]https://tech.ebu.ch/publications/sqamcd

Fig. 2. Percentages of the clipped samples for the selected input SDRs.

The restored signal $\hat{\mathbf{x}}$ is evaluated using the output SDR, which is computed as $\mathrm{SDR}(\mathbf{x}, \hat{\mathbf{y}})$ using (25). Note that evaluating the SDR on the whole signal may handicap the methods that produce signals inconsistent in the reliable part. (Some of such methods may include replacing the reliable samples with the reliable samples from the input clipped signal $\mathbf{y}$ as the final step of the reconstruction.) Therefore, as for example in [21], we compute the SDR on the clipped part only, SDR_c, as

$$\mathrm{SDR}_c(\mathbf{x}, \hat{\mathbf{x}}) = 20 \log_{10} \frac{\left\| \begin{bmatrix} M_H \\ M_L \end{bmatrix} \mathbf{x} \right\|_2}{\left\| \begin{bmatrix} M_H \\ M_L \end{bmatrix} \mathbf{x} - \begin{bmatrix} M_H \\ M_L \end{bmatrix} \hat{\mathbf{x}} \right\|_2}. \tag{26}$$

Since this article aims at signal reconstruction, we will rather use the SDR improvement, i.e., the difference between the SDR of the restored and the clipped signal, formally defined as

$$\Delta\mathrm{SDR}_c = \mathrm{SDR}_c(\mathbf{x}, \hat{\mathbf{x}}) - \mathrm{SDR}_c(\mathbf{x}, \mathbf{y}), \tag{27}$$

and similarly for $\Delta\mathrm{SDR}$. Note that this criterion does not take into consideration whether a method is or is not consistent in the clipped part, since (26) does not either. Note also that in the case of consistency in the reliable part, the $\Delta\mathrm{SDR}$ produces the same values, no matter whether the SDR is computed on the whole signal or on the clipped samples only, and $\Delta\mathrm{SDR}$ and $\Delta\mathrm{SDR}_c$ simply coincide.

2) PEAQ: A good quality assessment should correspond as much as possible to perceptual experience. From this point of view, the SDR is not the best metric, since the physical similarity with the original signal does not automatically imply perceptual similarity and quality. Hence, an evaluation metric involving the human perceptual system should be used.

PEAQ—Perceptual Evaluation of Audio Quality, which became the ITU-R recommendation (BS 1387) in 1999, is considered standard for audio quality evaluation. The final output of PEAQ is the Objective Difference Grade (ODG) rating the perceived difference between the clean and the degraded signal. The ODG score is shown in Table II.

For the experiments, we used the MATLAB implementation,[4] implemented according to the revised version of PEAQ (BS.1387-1) and available with a detailed review of this

[4]http://www-mmsp.ece.mcgill.ca/Documents/Software/

method [84]. Unfortunately, this implementation is suited only for signals with the 48 kHz sampling frequency. Since the audio database used is sampled at 44.1 kHz, we upsample the signals in order to compute the PEAQ ODG.

TABLE II
OBJECTIVE DIFFERENCE GRADE

ODG	Impairment description
0.0	Imperceptible
−1.0	Perceptible, but not annoying
−2.0	Slightly annoying
−3.0	Annoying
−4.0	Very annoying

3) PEMO-Q: As another evaluation metric taking into account the human auditory system we use the PEMO-Q method published in [85]. The Matlab implementation[5] is freely available for academic use and research. PEMO-Q computes the perceptual similarity measure (PSM), which can be mapped to the ODG score (see Table II). The mapping to ODG is available only for signals with 44.1 kHz sampling frequency but since this is the case of the database used, no additional resampling is required.

4) R_{nonlin}: Even though PEMO-Q and PEAQ provide a reasonable prediction of perceptual evaluation and are often used in restoration studies, they were actually designed for the evaluation of bit-rate reduction systems in audio coding. For this reason, we also include the R_{nonlin} metric [86], designed specifically for evaluating nonlinear distortions.

The R_{nonlin} metric can be used to predict the perceived quality of nonlinearly distorted signals. The prediction, which is a value on a scale from 0 to 1 (worst to best), is based on the weighted cross-correlation of the output of an array of gammatone filters. The Matlab implementation is available at Matlab central[6].

C. Algorithms and settings

Parameter fine-tuning is a necessary part of the experiment, as the overall results depend highly on the parameter selection. In our experiments, we attempted to tune the parameters of each algorithm such that it produces the best possible reconstruction result in terms of the SDR.

Several above-presented algorithms employ a time-frequency (TF) transform and / or processing by blocks. For such cases, we tried to unify the related setting across the algorithms to ensure a fair comparison. The optimal way would be to tune the parameters for every input SDR separately (for instance, harsh clipping with a great number of clipped samples benefits from the use of longer windows compared to mild clipping). For simplicity, we stick to a compromise among all the cases, and the parameters of each method stay fixed for all test signals and all clipping levels. Specifically, if the algorithm processes the signal block-by-block, 8192-sample-long (∼186 ms) blocks are used. If the algorithm utilizes the DGT, we use the 8192-sample-long Hann window with 75%

overlap and 16384 frequency channels. Unfortunately, such a setting could not be used in C-OMP, NMF and DL due to the high computational complexity of these algorithms. For C-OMP and DL, we use windows of 1024 samples, with 75% overlap, and twice-redundant dictionaries of size $P = 2048$. The NMF algorithm use windows of size 2048, with 2048 frequency channels.

The implementation of the TF transforms is handled by the LTFAT toolbox [38] in most of the methods.

As for the termination criterion, we resort to using just a simple maximum number of iterations. This number has been empirically set for each algorithm independently to make sure that the algorithm fully converged. The only exception is the SPADE algorithms that have the ϵ parameter involved straight in the problem formulation, see (5) and (7), and ϵ is thus naturally used as the termination threshold.

If an algorithm produces a signal inconsistent in the reliable part, we do not replace that part with the original reliable samples before the evaluation. Naturally, such a replacement increases the overall output SDR, as discussed in Sec. V-B. In terms of perceptual metrics, evidence suggests that doing this replacement is not beneficial for signals with low input SDR. This is probably due to the fact that the replacement of the samples creates discontinuities in the waveform, leading to the introduction of artificial higher harmonics that in the end degrade the reconstruction quality. The advantage of matching the reliable samples is, however, pronounced in the high input SDR regime, where there is only a small number of clipped samples and the effect of new discontinuities is negligible. Note that these additional variants are excluded from the comparison for clarity; however, the supplementary repository presents the described effect in the form of graphs.

1) Constrained OMP: C-OMP was tested using the implementation provided by the authors of [15], [44] in the SMALLbox MATLAB toolbox.[7] We have used the DGT-based implementation with min-constraint, as we have found that it provided a reasonable trade-off between performance and computational complexity. Note that when the clipping level is low (many samples missing), the constrained optimization (the final step of Alg. 1) often failed to converge. We believe this is because the support set Ω estimated with the OMP (without any clipping constraint) is often suboptimal, leading to a signal that is clipping-*inconsistent*. As a result, a signal that belongs to the range space of D_Ω and the clipping consistency set $\Gamma_{\mathrm{H}} \cap \Gamma_{\mathrm{L}}$ might not exist or have a very large amplitude, thus making the constraint $D_\Omega \mathbf{z}_\Omega \in \Gamma_{\mathrm{H}} \cap \Gamma_{\mathrm{L}}$ unfeasible. In that scenario, following the guidelines by the authors in [44] (implemented in the accompanying code), we simply return the output of the (unconstrained) OMP as a solution.

2) SPADE: Although we have the original implementation of A-SPADE, we use our own implementation, which is slightly improved compared to the original version. The main differences are the means by which the signal is windowed and the hard-thresholding step, where we take into account the complex conjugate structure of the DFT and always process pairs of coefficients (hence producing purely real signals with

---

the inverse DFT). The above also applies to the S-SPADE implementation.

Both SPADE algorithms process the signal by overlapping blocks. Each block is processed separately and the blocks are folded back using the standard overlap–add (OLA) procedure. The frequency representation in each block is computed using a twice redundant DFT (forming a Parseval frame). The parameters of S-SPADE and A-SPADE are identical and they correspond to the description in Sec. IV-B. It is fairly easy and intuitive to tune them. The algorithm works very well with default parameters ($s = 1$, $r = 1$). During testing, we found out that for the most extreme clipping (input SDR = 1 dB) it helps to increase the number of iterations slower, by incrementing k every even iteration, i.e., $r = 2$. This option lifted the average output SDR by 1.2 dB.

The termination criterion is based on the residue being minimized, i.e., on $A\mathbf{x} - \mathbf{z}$ for A-SPADE and on $\mathbf{x} - D\mathbf{z}$ for S-SPADE. The algorithms run until the ℓ_2 norm of the residue is smaller than ϵ. We use the default $\epsilon = 0.1$ used in the original papers. Decreasing ϵ may increase the number of iterations but does not improve the overall reconstruction quality (in none of the considered quality measures).

3) Plain ℓ_1 minimization: The algorithms based on ℓ_1 relaxation described in Sec. IV-D are designed to process the input signal all at once using the DGT, with the DGT parameters specified above in this section.

The synthesis variant was computed using the DR algorithm (Alg. 4) and the analysis variant was solved by the CP algorithm (Alg. 5). It was quite difficult to tune the parameters in these algorithms, since each test sound required a slightly different setting to obtain reasonable convergence. It also happened sometimes that the output SDR started to drop after reaching its peak, and then it stabilized at a lower value. Nevertheless, we kept the algorithms running for the full number of iterations, since usually the values of the objective metrics still got increased even after this SDR peak point.

In such a case, we let the algorithm reach the maximum number of iterations and we take the result from the final iteration. Note that the just-described behavior is more common for methods below that employ the same DR and CP algorithms but use coefficient weighting, i.e., $\mathbf{w} \neq \mathbf{1}$. Because of these issues, we set all the parameters to unity, which turns out to be a compromise covering all the cases; we set $\lambda = \gamma = 1$ for the DR algorithm and $\zeta = \sigma = \rho = 1$ for the CP algorithm.

In both algorithms, the convergence criterion was set strictly to 3000 iterations, where it was certain that the algorithms reached the minimum with sufficient accuracy.

4) Declipping in Sparseland $R\ell_1CC$: The original codes of the $R\ell_1CC$ method rely on the CVX, whose disadvantage is that the transforms are handled only in the form of a matrix; this prevents CVX from using longer window lengths. Therefore, we re-implemented the original approach using the DR algorithm, as described in Alg. 6. We no longer used matrices but fast operator-based transforms. For the DR algorithm, we used the same setting as in the non-weighted case, i.e., $\lambda = \gamma = 1$, but the maximum number of the DR iterations was set to 1000. The original paper [16] used 10 outer-cycle iterations of the $R\ell_1CC$ algorithm, but our implementation

used only 6, since after the sixth iteration the performance starts to decrease rapidly.

Concerning the operators, we used the DGT instead of the DFT used for toy examples in [16]. The parameter ϵ used for updating the weights was set to 0.0001. Besides that, we introduced another parameter, δ; the inner cycle is terminated if the relative change of the DGT coefficients between two subsequent iterations drops below δ. The specific value used in the tests was $\delta = 0.01$. This modification was used just to speed up the computations and has no effect on the reconstruction quality.

On top of that, we also include the analysis variant using the CP algorithm (with $\zeta = \sigma = \rho = 1$ and 1000 inner iterations). Also in case of the analysis variant, the reconstruction quality starts to decrease with the seventh outer iteration.

5) Social Sparsity: For the experiments, we used the implementation of the algorithm kindly provided by the authors of [21]. For clarity, only the best performing variants are included in the evaluation, i.e., Empirical Wiener (EW) and Persistent EW (PEW). In the case of the PEW, one needs to specify the size of the coefficient neighborhood in the TF plane. For our test case (audio at 44.1 kHz and the DGT), the best-performing size of the neighborhood was 3×7 (i.e., 3 coefficients in the direction of frequency and 7 coefficients in time).

One needs to carefully tune the number of inner and outer iterations and the distribution of the parameter λ during the course of iterations (see Sec. IV-F). In the final algorithm, we used 500 inner and 20 outer iterations with λ logarithmically decreasing from 10^{-1} to 10^{-4}.

As for the step size, the authors of [21] report using $\gamma = 0.9$, leading to fast convergence. Following the codes provided by the authors, our γ rather develops according to the formula $\frac{k-1}{k+5}$, where k is the iteration counter. Such an approach corresponds to acceleration in FISTA [69].

Sometimes it happens that the optimization gets stuck (especially in the first couple of outer iterations) and starts to converge again with the next outer iteration (i.e., when λ is decreased). For this reason, we slightly modified the implementation by adding the δ threshold, used to break the outer iteration even if the maximum number of inner iterations has not been reached. The ℓ_2 norm of the difference between the time-domain solutions of the current and previous iteration is compared with δ, whose value was set to 0.001.

Even though the algorithm does not produce signals consistent in any earlier defined sense, the result of social sparsity approaches is usually not too far from the set Γ.

6) Perceptual Compressed Sensing: We were able to obtain neither the implementation of this method nor any example of output signals, unfortunately. The authors say that it relies on CVX, which is not quite practical for our experiment. Since the scores reported in [19] looked promising, we re-implemented the method using the Condat-Vũ algorithm, which mimics the algorithmic scheme suggested in [19]. The signal is processed block by block as in the SPADE algorithms.

Our experiment includes the non-weighted variant, CSL1, the perceptually-weighted variant, PCSL1, and, inspired by very good results of quadratic weights in [33], an additional parabola-weighted variant, referred to as PWCSL1.

The parameters of the CV algorithm were set to $\gamma = 0.01$, $\sigma = 1$, $\tau \approx 0.0186$, and $\rho = 0.99$ for all three mentioned variants. The maximum number of iterations was set to 500 for CSL1 and PCSL1 and 5000 for PWCSL1 (convergence is significantly slower in the latter case).

In contrast to the original article, the "official" implementation of MPEG PM 1 could not be used because it is strictly limited to 512-sample-long windows. In this survey, our goal is to compare algorithms with the best possible settings and with the same DGT settings across all methods. Therefore, we wanted to work with a Hann window 8192 samples long with 75 % overlap and 16384 frequency channels. Hence, instead of the official implementation, we had to switch to a slightly modified and simplified version of MPEG Layer 1 PM 1, which is not restricted in terms of the block length.

The PCSL1 algorithm places the original reliable samples at the reliable positions at the very end of processing. This leads to some ODG gain for mild clipping levels. However, as discussed earlier, we present results without such a replacement.

7) Psychoacoustically motivated ℓ_1 minimization: The algorithms used here are the same as in the case of the plain ℓ_1 minimization (Sec. V-C3), but the weights are now perceptually motivated. The original paper [33] presented several ways of implementing the psychoacoustical information into the declipping process (only for the synthesis case). The best option turned out to be simple weights that grow quadratically with frequency.

In the experiments, we present only this "parabola-weighted" option, but we newly include the analysis variant as well. All the parameters and the maximum number of iterations for both DR and CP are identical to the plain case.

8) Dictionary Learning approach: For the dictionary learning algorithm, each signal is first decomposed into T overlapping windows $\mathbf{y}_1, \ldots, \mathbf{y}_T$ of size 1024, for a total of approximately $T = 1200$ windows per signal. These are directly used as inputs to the algorithm, such that the dictionary is learned and evaluated on the same signal.

As the optimal sparsity parameter K depends on the signal as well as on the clipping level, we adopt here an *adaptive sparsity* strategy. In the first sparse coding step (20), we first iterate (20) with $K = 1$, then sequentially increase K every few iterations, until an error threshold ϵ is reached. The resulting sparsity level $\hat{K}$ is then fixed throughout the rest of the algorithm. The dictionary algorithm is initialized with a real DCT dictionary of size $P = 2048$. We perform 20 iterations of sparse coding and dictionary update steps. The sparse coding steps (apart from the first one, which uses an adaptive sparsity strategy) are computed using 20 iterations of consistent IHT. The dictionary update step is computed using 20 steps of accelerated gradient descent. To improve the convergence, the sparse coefficients and dictionary are always initialized using estimates from the previous iteration. Using the learned dictionary $\hat{D}$ and sparse coefficients $\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_T$, each of the individual windows is reconstructed as $\hat{D}\hat{\mathbf{z}}_1, \ldots, \hat{D}\hat{\mathbf{z}}_T$, and the estimated signal is then recovered using the overlap–add.

9) NMF: As for the NMF-based declipping, we used the STFT with the sine window of size $F = 2048$ samples with 50% overlap. The NMF model order was set to $K = 20$. The GEM algorithm 10 was run for 50 iterations. These choices follow those used in the corresponding paper [24] except for the STFT window size, which was 64 ms (1024 samples for 16kHz-sampled signals) in [24] and is 46.4 ms (2048 samples for 44.1kHz-sampled signals) here. Note that the computational load of the NMF approach grows drastically with increasing window length. This is due to matrix inversion in Wiener filtering (24) and to the iterative ad hoc clipping constraint management strategy described at the end of Sec. IV-J. As such, even with the 2048-sample-long window, declipping some sequences took more than 8 hours. This is why we have not chosen a longer window size for experiments.

10) Janssen: For the evaluation of the Janssen algorithm, we have adapted the codes published in the SMALLbox. The signal is processed block by block. The order of the AR filter does not play a significant role in the quality of declipping; it turns out that the number of iterations is significantly more important. The ΔSDR increases with the number of iterations, but from a certain point, the algorithm starts to diverge wildly. The reason is probably that the algorithm does not have enough reliable data to fit reasonable AR parameters to the known signal. This point of change differs among the test signals and it is highly influenced by the input SDR (the smaller the input SDR, the sooner the divergence appears). In our experiments, the order of the filter is set to 512 and we run 3 iterations. Using 5 iterations produces slightly better results, but for one of the test signals at 1 dB input SDR the divergence appears, devaluating the average score.

D. Computational cost

Though the survey concentrates primarily on the reconstruction quality, some remarks on the computational cost of the methods are valuable. Below we quote the rough computational time needed to process one second of audio (i.e., 44100 samples). No parallel computing is considered, and the time spent by tuning the parameters is not included. The overall time needed to completely recompute our experiments is estimated at one month when performed on a common PC.

1) C-OMP: It takes 5 to 10 minutes to declip one second of the signal.
2) SPADE: The clipping threshold determines the performance—the higher the input SDR, the longer it takes for the SPADE algorithms to converge. Processing one second of audio takes from 22 up to 64 seconds for A-SPADE and 14 up to 52 seconds for S-SPADE.
3) Plain ℓ_1: Computational time is roughly 20 seconds for both the DR and CP algorithms, independent of the clipping threshold.
4) Reweighted ℓ_1: It takes 66 seconds for the DR algorithm and 56 seconds for the CP algorithm.
5) Social Sparsity: Slightly less than 2 minutes for EW and slightly more than 2 minutes in the case of PEW.
6) Perceptual Compressed Sensing: CSL1 & PCSL1: Roughly 20 seconds, PWCSL1 below three minutes.
7) PWℓ_1: Same as for Plain ℓ_1.

8) Dictionary Learning: 1 to 2 minutes depending on the clipping level, the algorithm generally converging a bit faster when the clipping level is low.

9) NMF: Average computation time is 30 minutes per one second of audio. The particular time depends on the input SDR—for the lowest, the cost can rise up to 1 hour.

10) Janssen: Depends heavily on the input SDR; the durations differ by two orders of magnitude: for 1 dB input SDR Janssen takes about 16 minutes per second of audio, and for 20 dB input SDR it is done in 5–15 s.

E. Results and discussion

Results of the declipping in terms of performance are presented and commented on in this section. Recall that the comparison is done in terms of four objective metrics—ΔSDR_c, PEAQ, PEMO-Q and R_{nonlin}. In the bar graphs that follow, algorithms coming from the same family share the same color. If a method was examined in both the analysis and the synthesis variant, the analysis variant is graphically distinguished via hatching. Other variants (e.g., multiple shrinkage operators in the SS algorithms or different weights within the CSL family) use gray stippling. The abbreviations used in the legends are used all over the text, but also summarized in Table III.

The ΔSDR_c results are presented in Fig. 3, PEAQ ODG values in Fig. 4, PEMO-Q ODG values in Fig. 5, and R_{nonlin} values in Fig. 6. The reader can easily draw a number of conclusions by studying the plots; however, we try to summarize the most important and interesting facts inferred from these results. We concentrate more on the ODG score designed to reflect properties of the human auditory system.

First of all, note that the SDR scores correlate with the ODG scores to a certain degree. There are exceptions, however—compare, for example, Rℓ_1CC CP and Rℓ_1CC DR, which behave just the opposite way (SDR versus ODG).[8] Note also that the ODG values of PEMO-Q are uniformly worse than those of PEAQ, but the relation between scores of the individual methods is retained. An exception is the family of CSL1, PCSL1, PWCSL1, where we observe a difference.

The main messages can be summarized as follows:

- Both variants of the SPADE algorithm perform similarly and very well in terms of all the four metrics and across all levels of degradation, while the analysis variant is slightly preferred.
- Using social sparsity leads to very good results. In particular, the SS PEW method (which assumes persistence of frequencies in time) performs overall best in terms of the SDR and is one of the best few in terms of the perceptual measures.
- In the medium to mild clipping regime, NMF is the clear winner in terms of ODG and it also performs very good in SDR. With more severe clipping (1 and 3 dB) it behaves worse but is still very competitive.
- Despite its simplicity, the results of the parabola-weighted ℓ_1 minimization are uniformly very good, in terms of all

the metrics. Its SDR values more or less correspond to those of SPADE, and the ODG scores show that for wild clipping, PWℓ_1 is even the best declipping method.

- Introduction of reweighting improves the plain ℓ_1 minimization, especially for the analysis variant, but this observation only holds for the SDR. In terms of ODG, the effect is reversed. In fact, the performance of plain ℓ_1 can be found surprisingly satisfactory in the PEAQ ODG graph.
- The family of psychoacoustically weighted optimizations (CSL1) failed. The best results are achieved using the parabolic weights, which are in fact not psychoacoustically motivated. These observations are especially interesting since the original article [19] reported a better declipping quality (but on a different dataset).
- The low scores of dictionary learning may be probably attributed to the fact that it uses the IHT algorithm in its sparse coding step (see Alg. 9), which has been recently surpassed by its successor, SPADE, for example. Another factor could be that the initial dictionary is the real-valued DCT and that the iterates of Alg. 9 remain in the real domain, causing phase-related artifacts.
- The Janssen method performs well only in the very high input SDR regime, otherwise it fails due to the lack of reliable samples. Recall that Janssen was included in the study since it performs on par with state-of-the-art methods in audio inpainting (of compact signal gaps). Clearly, the hypothesis that it could be similarly successful also in declipping is not validated.

Besides the reconstruction quality, which is the main concern of the article, other factors can also be taken into consideration. For example, consider the rough cost of computation that has been reported in Sec. V-D. According to these values, the NMF is 90-times slower than the PWℓ_1 algorithms, while producing results of almost identical quality for most clipping levels. On the other hand, some methods require painful parameter tuning to achieve good results. In that regard, NMF or SPADE in particular can be seen as advantageous.

F. Software and data

Besides the numerical evaluation, the intention of the article is to collect implementations of the examined methods and to make them publicly available, both for the reproducibility purposes and to stimulate future research in this area. The webpage

https://rajmic.github.io/declipping2020/

contains the link to the MATLAB implementation (except the NMF, which is not publicly available). The tests have been performed in MATLAB version 2019b.

This article provides an objective evaluation through PEAQ and PEMO-Q, which pursue being as close as possible to human perception. Individual subjective assessment, which is always the most decisive, can be made via the supplied webpage, where all sound examples are directly playable (or downloadable).

---

[8]This corresponds to our experience with reweighting in audio inpainting, see [64], but we do not have an explanation for this effect unfortunately.
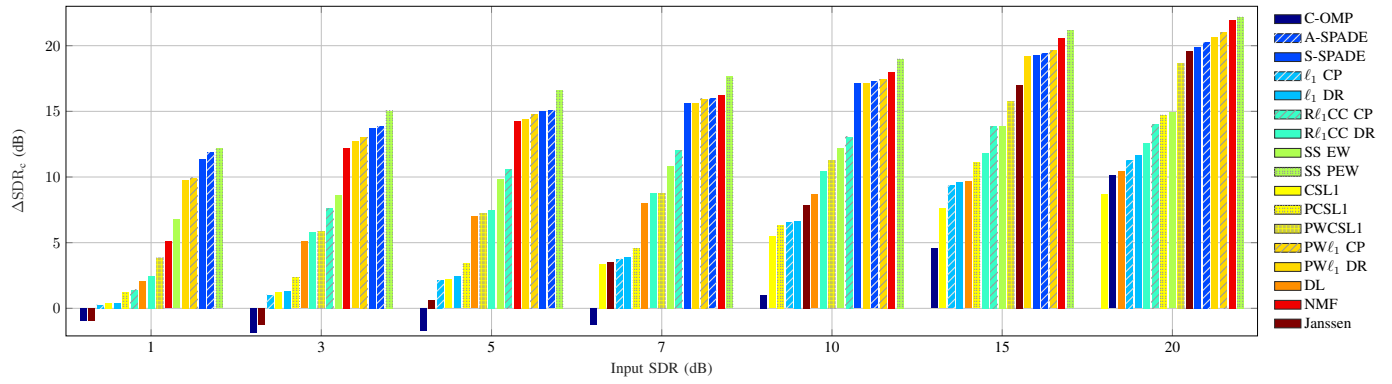
Fig. 3. Average $\Delta\mathrm{SDR_c}$ results. The abbreviations from the legend are summarized in Table III.
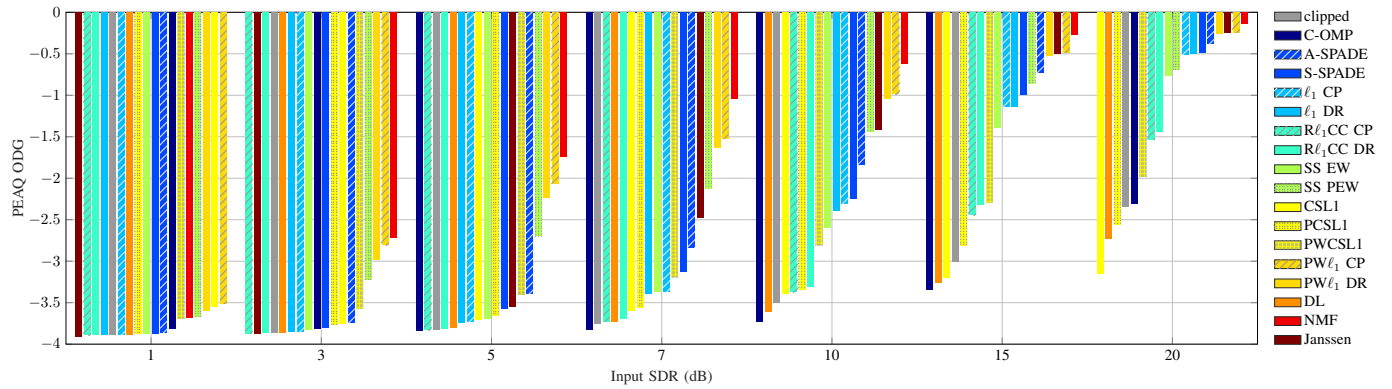

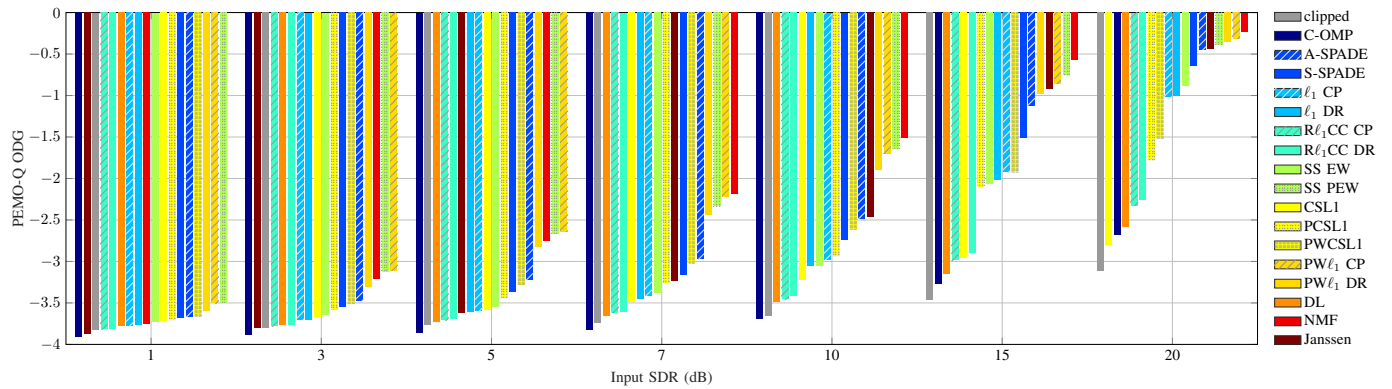Fig. 4. Average PEAQ ODG results. The PEAQ ODG of the clipped signal is depicted in gray.
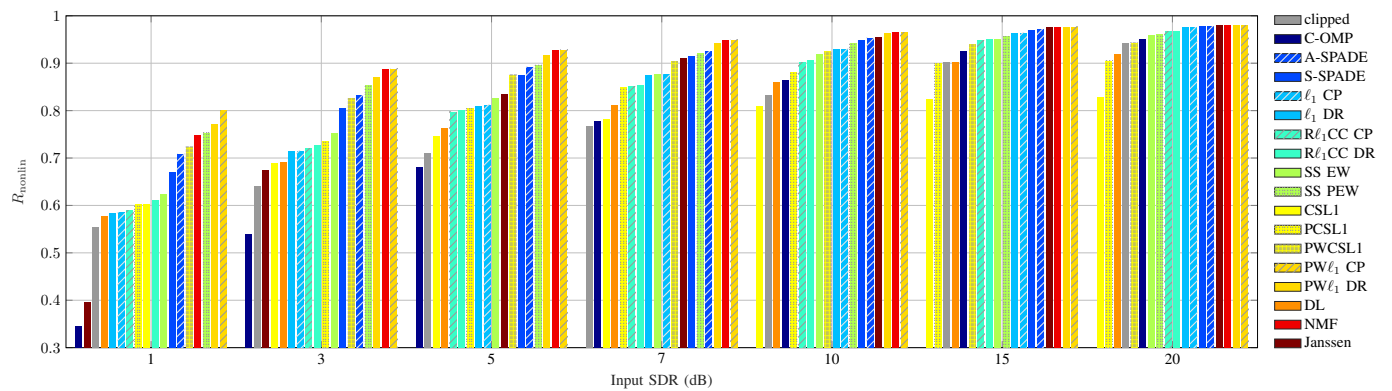

Fig. 5. Average PEMO-Q ODG Results.

Fig. 6. Average R_{nonlin} Results.

TABLE III
TABLE OF EXAMINED ALGORITHMS AND THEIR ABBREVIATIONS AND REFERENCES. SOME OF THE ALGORITHMS WERE PROPOSED IN THIS ARTICLE IN ORDER TO MAKE IT AS COMPLETE AS POSSIBLE. TWO ALGORITHMS ARE NOT PRESENTED.

Abbreviation	Full name	Algorithm utilized	Reference
C-OMP	Constrained Orthogonal Matching Pursuit	Alg. 1	Adler'11 [15]
S-SPADE	Synthesis SParse Audio DEclipper	Alg. 3	Záviška'19 [32]
A-SPADE	Analysis SParse Audio DEclipper	Alg. 2	Kitić'15 [25]
ℓ_1 DR	ℓ_1 minimization using Douglas–Rachford (synthesis)	Alg. 4	Rajmic'19 [59]
ℓ_1 CP	ℓ_1 minimization using Chambolle–Pock (analysis)	Alg. 5	analysis variant of [59]
Rℓ_1CC DR	Reweighted ℓ_1 min. with Clipping Constraints using Douglas–Rachford (synthesis)	Alg. 6	Weinstein'11 [16]
Rℓ_1CC CP	Reweighted ℓ_1 min. with Clipping Constraints using Chambolle–Pock (analysis)	—	analysis variant of [16]
SS EW	Social Sparsity with Empirical Wiener	Alg. 7	Siedenburg'14 [21]
SS PEW	Social Sparsity with Persistent Empirical Wiener	Alg. 7	Siedenburg'14 [21]
CSL1	Compressed Sensing method minimizing ℓ_1 norm	Alg. 8	Defraene'13 [19]
PCSL1	Perceptual Compressed Sensing method minimizing ℓ_1 norm	Alg. 8	Defraene'13 [19]
PWCSL1	Parabola-Weighted Compressed Sensing method minimizing ℓ_1 norm	Alg. 8	variant of [19]
PWℓ_1 DR	Parabola-Weighted ℓ_1 minimization using Douglas–Rachford (synthesis)	Alg. 4	Záviška'19b [33]
PWℓ_1 CP	Parabola-Weighted ℓ_1 minimization using Chambolle–Pock (analysis)	Alg. 5	analysis variant of [33]
DL	Dictionary Learning approach	Alg. 9	Rencker'18 [29]
NMF	Nonnegative Matrix Factorization	Alg. 10	Bilen'15 [24]
Janssen	Janssen method for inpainting	—	Janssen'86 [11]

VI. CONCLUSION AND PERSPECTIVES

The article presented the declipping problem and an overview of methods used to solve it. Besides such a survey, several popular declipping methods of different kinds were selected for deeper evaluation. This is the first time so many methods are compared based on the same audio dataset (moreover sampled at 44.1 kHz). The main focus of the article is on the reconstruction quality, which is reported in terms of four metrics. However, other factors like the computation cost and complexity in tuning parameters are also discussed.

The algorithms studied and compared in this paper exhibit various performances and computational times. Some algorithms perform better at low clipping levels, while others perform better at high clipping levels. The choice of an algorithm thus depends on the input data. Nevertheless, the methods based on social shrinkage, nonnegative matrix factorization, weighting the transform coefficients and, last but not least, the SPADE seem to yield results that make them preferred choices; in most conditions, they scored significantly better than the majority of other methods in terms of the perceptual metrics. Depending on the application, the computational time of each algorithm and the time-consuming tuning of parameters might also be a decisive selection criterion. From these viewpoints, variants of the SPADE are attractive.

Directions for future research may include combining strategies and modeling assumptions of the various algorithms presented in this paper. For instance, the social sparsity regularizer of [21] or the perceptual weights of [19] could be combined with S-SPADE or other algorithms. Dictionary learning could be combined with S-SPADE or social sparsity. Most algorithms discussed here use the synthesis model, whiledeveloping their *analysis* counterpart could also be a promising idea. We could also imagine assigning weights, in order to favor clipping consistency. Finally, we have focused in this paper on unsupervised techniques, which do not rely on a training set with clean data. However, the success of supervised techniques, and in particular deep learning based techniques, in tackling many other problems in computer vision, speech recognition, and audio analysis, motivates further study of supervised techniques in audio declipping. Recent deep learning approaches to audio declipping have shown promising results in the context of speech declipping [6], [7], [8]. A potential research direction would be to combine the power of supervised techniques with signal assumptions, modeling and algorithms discussed in this article. One of such directions could be the recent finding that artificial networks that bear the structure of unfolded proximal algorithms are able to join the signal modeling and learning from data, possibly keeping advantages of both distinct worlds, see for example [87] in the context of image processing.

Note that this survey and the papers under consideration only investigate declipping of signals that are clipped in the digital domain. However, when clipping occurs in the analog domain, it is different since before the A/D conversion, a low-pass filter is applied to avoid aliasing. Since the clipping distortion is wide-band, clipping aliasing [88], quite an unpleasant distortion, is present in the latter case. This effect is well-known to audio engineers, and a digital aliasing-free clipping or compression of dynamics is often performed via upsampling [89]. For example, [90] addresses the aliasing reduction in digitally-clipped signals, though without declipping itself. While the methods covered by this survey reduce both aliasing and the remaining narrow-band clipping distortion, if the clipping is carried out in the analog domain or in a particular aliasing-free manner (e.g., via upsampling [89]), different new declipping algorithms should be developed and applied, simply because the clipping process is different in that case.

REFERENCES

[1] C.-T. Tan, B. C. J. Moore, and N. Zacharov, "The effect of nonlinear distortion on the perceived quality of music and speech signals," *Journal of the Audio Engineering Society*, vol. 51, no. 11, pp. 1012–1031, 2003.

[2] J. Málek, "Blind compensation of memoryless nonlinear distortions in sparse signals," in *2013 21st European Signal Processing Conference (EUSIPCO)*, Sept. 2013.

[3] Y. Tachioka, T. Narita, and J. Ishii, "Speech recognition performance estimation for clipped speech based on objective measures," *Acoustical Science and Technology*, vol. 35, no. 6, pp. 324–326, 2014.

[4] A. Ozerov and C. Févotte, "Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 550–563, 2009.

[5] R. Sathya and A. Abraham, "Comparison of supervised and unsupervised learning algorithms for pattern classification," *Intl. Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 2, pp. 34–38, 2013.

[6] F. Bie, D. Wang, J. Wang, and T. F. Zheng, "Detection and reconstruction of clipped speech for speaker recognition," *Speech Communication*, vol. 72, pp. 218–231, 2015.

[7] H. B. Kashani, A. Jodeiri, M. M. Goodarzi, and S. G. Firooz, "Image to image translation based on convolutional neural network approach for speech declipping," in *4th Conference on Technology In Electrical and Computer Engineering (ETECH 2019)*, Apr. 2019.

[8] W. Mack and E. A. P. Habets, "Declipping speech using deep filtering," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct. 2019, pp. 200–204.

[9] A. Ozerov, Ç. Bilen, and P. Pérez, "Multichannel audio declipping," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2016, pp. 659–663.

[10] C. Gaultier, N. Bertin, and R. Gribonval, "Cascade: Channel-aware structured cosparse audio declipper," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018, pp. 571–575.

[11] A. J. E. M. Janssen, R. N. J. Veldhuis, and L. B. Vries, "Adaptive interpolation of discrete-time signals that can be modeled as autoregressive processes," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 2, pp. 317–330, 1986.

[12] J. Abel and J. Smith, "Restoring a clipped signal," in *1991 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 1991, pp. 1745–1748, vol. 3.

[13] W. Fong and S. Godsill, "Monte carlo smoothing for non-linearly distorted signals," in *2001 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2001, pp. 3997–4000, vol. 6.

[14] A. Dahimene, M. Noureddine, and A. Azrar, "A simple algorithm for the restoration of clipped speech signal," *Informatica*, vol. 32, pp. 183–188, 2008.

[15] A. Adler, V. Emiya, M. Jafari, M. Elad, R. Gribonval, and M. Plumbley, "A constrained matching pursuit approach to audio declipping," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 329–332.

[16] A. J. Weinstein and M. B. Wakin, "Recovering a clipped signal in sparseland," *Sampling Theory in Signal and Image Processing*, vol. 12, no. 1, pp. 55–69, 2013.

[17] S. Miura, H. Nakajima, S. Miyabe, S. Makino, T. Yamada, and K. Nakadai, "Restoration of clipped audio signal using recursive vector projection," in *2011 IEEE Region 10 Conference (TENCON)*, Nov. 2011, pp. 394–397.

[18] S. Kitić, L. Jacques, N. Madhu, M. Hopwood, A. Spriet, and C. De Vleeschouwer, "Consistent iterative hard thresholding for signal declipping," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2013, pp. 5939–5943.

[19] B. Defraene, N. Mansour, S. D. Hertogh, T. van Waterschoot, M. Diehl, and M. Moonen, "Declipping of audio signals using perceptual compressed sensing," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 12, pp. 2627–2637, 2013.

[20] I. Selesnick, "Least squares with examples in signal processing," Apr. 2013. [Online]. Available: https://cnx.org/contents/XRPKcVgh@1/Least-Squares-with-Examples-in-Signal-Processing.

[21] K. Siedenburg, M. Kowalski, and M. Dorfler, "Audio declipping with social sparsity," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 1577–1581.

[22] S. Kitić, N. Bertin, and R. Gribonval, "Audio declipping by cosparse hard thresholding," in *2nd Traveling Workshop on Interactions between Sparse models and Technology*, Aug. 2014.

[23] M. Jonscher, J. Seiler, and A. Kaup, "Declipping of speech signals using frequency selective extrapolation," in *Speech Communication; 11. ITG Symposium*, Sept. 2014, pp. 1–4.

[24] Ç. Bilen, A. Ozerov, and P. Pérez, "Audio declipping via nonnegative matrix factorization," in *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct. 2015, pp. 1–5.

[25] S. Kitić, N. Bertin, and R. Gribonval, "Sparsity and cosparsity for audio declipping: a flexible non-convex approach," in *12th International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, Aug. 2015, pp. 243–250.

[26] M. J. Harvilla and R. M. Stern, "Efficient audio declipping using regularized least squares," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2015, pp. 221–225.

[27] T. Takahashi, K. Uruma, K. Konishi, and T. Furukawa, "Block adaptive algorithm for signal declipping based on null space alternating optimization," *IEICE Transactions on Information and Systems*, vol. E98.D, no. 1, pp. 206–209, 2015.

[28] F. Elvander, J. Swärd, and A. Jakobsson, "Grid-less estimation of saturated signals," in *2017 51st Asilomar Conference on Signals, Systems, and Computers (ACSSC)*, Oct. 2017, pp. 372–376.

[29] L. Rencker, F. Bach, W. Wang, and M. D. Plumbley, "Consistent dictionary learning for signal declipping," in *14th International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, July 2018, pp. 446–455.

[30] G. Chantas, S. Nikolopoulos, and I. Kompatsiaris, "Sparse audio inpainting with variational bayesian inference," in *2018 IEEE International Conference on Consumer Electronics (ICCE)*, Jan. 2018, pp. 1–6.

[31] C. Gaultier, "Design and evaluation of sparse models and algorithms for audio inverse problems," Theses, Université Rennes 1, Jan. 2019.

[32] P. Záviška, P. Rajmic, O. Mokrý, and Z. Průša, "A proper version of synthesis-based sparse audio declipper," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 591–595.

[33] P. Záviška, P. Rajmic, and J. Schimmel, "Psychoacoustically motivated audio declipping based on weighted l1 minimization," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, July 2019, pp. 338–342.

[34] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming," http://cvxr.com/cvx.

[35] O. Christensen, *Frames and Bases, An Introductory Course*, Boston: Birkhäuser, 2008.

[36] M. Elad, P. Milanfar, and R. Rubinstein, "Analysis versus synthesis in signal priors," in *Inverse Problems*, vol. 23, no. 3, pp. 947–968, 2005.

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

[38] Z. Průša, P. L. Søndergaard, N. Holighaus, C. Wiesmeyr, and P. Balazs, "The Large Time-Frequency Analysis Toolbox 2.0," in *Sound, Music, and Motion*, Lecture Notes in Computer Science, Dec. 2014, pp. 419–442.

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

[40] P. Combettes and J. Pesquet, "Proximal splitting methods in signal processing," *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pp. 185–212, 2011.

[41] L. Condat, "A generic proximal algorithm for convex optimization—application to total variation minimization," *IEEE Signal Processing Letters*, vol. 21, no. 8, pp. 985–989, Aug 2014.

[42] M. Fadili and J.-L. Starck, "Monotone operator splitting for optimization problems in sparse recovery," IEEE Publishing, 2009, pp. 1461–1464.

[43] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[44] A. Adler, V. Emiya, M. Jafari, M. Elad, R. Gribonval, and M. Plumbley, "Audio Inpainting," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 3, pp. 922–932, 2012.

[45] T. Takahashi, K. Konishi, and T. Furukawa, "Hankel structured matrix rank minimization approach to signal declipping," in *2013 21st European Signal Processing Conference (EUSIPCO)*, Sept. 2013.

[46] C. Gaultier, S. Kitić, R. Gribonval, and N. Bertin, "Sparsity-based audio declipping methods: overview, new algorithms, and large-scale evaluation," arXiv:2005.10228 [cs.SD], May 2020.

[47] F. R. Ávila, M. P. Tcheou, and L. W. P. Biscainho, "Audio soft declipping based on constrained weighted least squares," *IEEE Signal Processing Letters*, vol. 24, no. 9, pp. 1348–1352, 2017.

[48] F. R. Ávila and L. W. P. Biscainho, "Audio soft declipping based on weighted l1-norm," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct. 2017, pp. 299–303.

[49] S. Gorlow and J. D. Reiss, "Model-based inversion of dynamic range compression," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 7, pp. 1434–1444, 2013.

[50] O. Mokrý, P. Záviška, P. Rajmic, and V. Veselý, "Introducing SPAIN (SParse Audio INpainter)," in *2019 27th European Signal Processing Conference (EUSIPCO)*, Sept. 2019.

[51] Y. Bahat, Y. Y. Schechner, and M. Elad, "Self-content-based audio inpainting," *Signal Processing*, vol. 111, pp. 61–72, 2015.

[52] A. Marafioti, P. Majdak, N. Holighaus, and N. Perraudin, "GACELA – A generative adversarial context encoder for long audio inpainting of music," *IEEE Journal of Selected Topics in Signal Processing*, 2020.

[53] J. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[54] B. L. Sturm and M. G. Christensen, "Comparison of orthogonal matching pursuit implementations," in *2012 20th European Signal Processing Conference (EUSIPCO)*, Aug. 2012, pp. 220–224.

[55] O. Mokrý and P. Rajmic, "Audio inpainting: Revisited and reweighted," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2906–2918, 2020.

[56] P. Záviška, O. Mokrý, and P. Rajmic, "S-SPADE Done Right: Detailed Study of the Sparse Audio Declipper Algorithms," Brno University of Technology, arXiv:1809.09847 [math.OC], Sept. 2018.

[57] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization," *Proceedings of The National Academy of Sciences*, vol. 100, no. 5, pp. 2197–2202, 2003.

[58] P. Combettes and J. Pesquet, "A Douglas–Rachford splitting approach to nonsmooth convex variational signal recovery," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 564–574, 2007.

[59] P. Rajmic, P. Záviška, V. Veselý, and O. Mokrý, "A new generalized projection and its application to acceleration of audio declipping," *Axioms*, vol. 8, no. 3, 2019.

[60] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *Journal of Mathematical Imaging and Vision*, vol. 40, no. 1, pp. 120–145, 2011.

[61] E. J. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *Journal of Fourier Analysis and Applications*, vol. 14, pp. 877–905, 2008.

[62] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical learning with sparsity: The lasso and generalization*, Boca Raton: CRC Press, 2015.

[63] P. Rajmic, "Exact risk analysis of wavelet spectrum thresholding rules," in *2003 10th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Dec. 2003, pp. 455–458, vol. 2.

[64] O. Mokrý and P. Rajmic, "Reweighted l1 minimization for audio inpainting," in *2019 Signal Processing with Adaptive Sparse Structured Representations (SPARS)*, July 2019.

[65] M. Novosadová and P. Rajmic, "Piecewise-polynomial signal segmentation using reweighted convex optimization," in *2017 40th International Conference on Telecommunications and Signal Processing (TSP)*, July 2017, pp. 769–774.

[66] M. Kowalski, K. Siedenburg, and M. Dörfler, "Social Sparsity! Neighborhood Systems Enrich Structured Shrinkage Operators," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2498–2511, 2013.

[67] I. Bayram, "Mixed norms with overlapping groups as signal priors," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 4036–4039.

[68] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[69] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[70] B. O'Donoghue and E. Candes, "Adaptive restart for accelerated gradient schemes," *Foundations of Computational Mathematics*, vol. 15, pp. 715–732, 2013.

[71] H.-Y. Gao, "Wavelet shrinkage denoising using the non-negative garrote," *Journal of Computational and Graphical Statistics*, vol. 7, no. 4, pp. 469–488, 1998.

[72] A. Antoniadis, "Wavelet methods in statistics: Some recent developments and their applications," *Statistics Surveys*, vol. 1, pp. 16–55, 2007.

[73] R. Gribonval and M. Nikolova, "A characterization of proximity operators," *Journal of Mathematical Imaging and Vision*, vol. 62, pp. 773–789, 2020.

[74] A. Spanias, T. Painter, and V. Atti, Audio Signal Processing and Coding, Wiley-Interscience, 2007.

[75] S. Shlien, "Guide to mpeg-1 audio standard," *IEEE Transactions on Broadcasting*, vol. 40, no. 4, pp. 206–218, 1994.

[76] B. C. Vũ, "A splitting algorithm for dual monotone inclusions involving cocoercive operators," *Advances in Computational Mathematics*, vol. 38, no. 3, pp. 667–681, 2013.

[77] J. Mairal, F. Bach, and J. Ponce, "Sparse modeling for image and vision processing," *Foundations and Trends® in Computer Graphics and Vision*, vol. 8, no. 2–3, pp. 85–283, 2014.

[78] L. Rencker, F. Bach, W. Wang, and M. D. Plumbley, "Sparse recovery and dictionary learning from nonlinear compressive measurements," *IEEE Transactions on Signal Processing*, vol. 67, no. 21, pp. 5659–5670, 2019.

[79] Ç. Bilen, A. Ozerov, and P. Pérez, "Solving time-domain audio inverse problems using nonnegative tensor factorization," *IEEE Transactions on Signal Processing*, vol. 66, no. 21, pp. 5604–5617, 2018.

[80] C. Févotte and M. Kowalski, "Estimation with low-rank time–frequency synthesis models," *IEEE Transactions on Signal Processing*, vol. 66, no. 15, pp. 4121–4132, 2018.

[81] C. Févotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis," *Neural computation*, vol. 21, no. 3, pp. 793–830, 2009.

[82] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[83] L. Oudre, "Interpolation of missing samples in sound signals based on autoregressive modeling," *Image Processing On Line*, vol. 8, pp. 329–344, 2018.

[84] P. Kabal, "An examination and interpretation of ITU-R BS.1387: Perceptual evaluation of audio quality," MMSP Lab Technical Report, McGill University, May 2002.

[85] R. Huber and B. Kollmeier, "PEMO-Q—A new method for objective audio quality assessment using a model of auditory perception," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 6, pp. 1902–1911, 2006.

[86] C.-T. Tan, B. C. J. Moore, N. Zacharov, and V.-V. Mattila, "Predicting the perceived quality of nonlinearly distorted music and speech signals," *Journal of the Audio Engineering Society*, vol. 52, no. 7/8, pp. 699–711, 2004.

[87] J. H. R. Chang, C.-L. Li, B. Poczos, B. V. K. V. Kumar, and A. C. Sankaranarayanan, "One network to solve them all — Solving linear inverse problems using deep projection models," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 5889–5898.

[88] P. H. Kraght, "Aliasing in digital clippers and compressors," *Journal of the Audio Engineering Society*, vol. 48, no. 11, pp. 1060–1065, 2000.

[89] D. Mapes-Riordan, "A worst-case analysis for analog-quality (alias-free) digital dynamics processing," in *AES Convention 105*, Sept. 1998.

[90] F. Esqueda, S. Bilbao, and V. Välimäki, "Aliasing reduction in clipped signals," *IEEE Transactions on Signal Processing*, vol. 64, no. 20, pp. 5255–5267, 2016.