Sparse Pursuit and Dictionary Learning for Blind Source Separation in Polyphonic Music Recordings

Sören Schulze, Emily J. King

***Abstract*—We propose a novel method for the blind separation of single-channel audio signals produced by the mixed sounds of musical instruments. While the approach of applying non-negative matrix factorization (NMF) has been studied in many papers, it does not make use of the pitch-invariance that the sounds of many instruments exhibit. This limitation can be overcome by using tensor factorization, in which context the use of log-frequency spectrograms was initiated, but this still requires the specific tuning of the instruments to be hard-coded into the algorithm. We develop a general-purpose sparse pursuit method that matches a discrete spectrum with given shifted continuous patterns. We first use it in order to transform our audio signal into a log-frequency spectrogram that shares properties with the mel spectrogram but is applicable to a wider frequency range. Then, we use the same algorithm to identify patterns from instrument sounds in the spectrogram. The relative amplitudes of the harmonics are saved in a dictionary, which is trained via a modified version of Adam. For a realistic monaural piece with acoustic recorder and violin, we achieve qualitatively good separation with a signal-to-distortion ratio (SDR) of 13.7 dB, a signal-to-interference ratio (SIR) of 28.1 dB, and a signal-to-artifacts ratio (SAR) of 13.9 dB, averaged over the instruments.**

***Index Terms*—Blind source separation, unsupervised learning, dictionary learning, pitch-invariance, pattern matching, sparsity, stochastic optimization, Adam, orthogonal matching pursuit**

I. Introduction

A. Problem Definition and Approach

THE *source separation* problem concerns itself with the recovery of signals $X_1, \ldots, X_c$ from a mixture $X = X_1 + \ldots + X_c$. We speak of *blind separation* when no specific prior information to characterize the sources of the signals is provided, especially not in the form of labeled training data.

Dictionary learning is the process of computing a collection of *atoms* whose linear combination is used to approximate the target data. This technique is very popular for the blind separation of audio sources, both of music recordings as well as of speech signals.

In order to apply dictionary learning on audio data, it is helpful to regard a time-frequency representation (*spectrogram*), which subdivides the problem into smaller time frames and highlights the frequency characteristics of the signal. One simple spectrogram is obtained via the modulus of the short-time Fourier transform (STFT) (cf. [1]). However, in the STFT spectrogram, different pitch manifests in linear scaling of the distances between the peaks in the frequency axis, which is computationally hard to handle.

Thus, we apply a novel *sparse pursuit* algorithm which identifies the tones in the STFT spectrogram and places them in a new spectrogram with a logarithmic frequency axis. On this, we apply the dictionary learning algorithm, which alternates between a sparse pursuit and a dictionary update step. Once the final dictionary has been computed, we apply the sparse pursuit algorithm on the entire log-frequency spectrogram in order to obtain the separated spectrograms, and after masking with the original mixture spectrogram, we employ the algorithm by Griffin and Lim [2] in order to convert them back into time-domain signals, using the phase of the original spectrogram as the initial value.

The overall procedure is displayed in Figure 1.

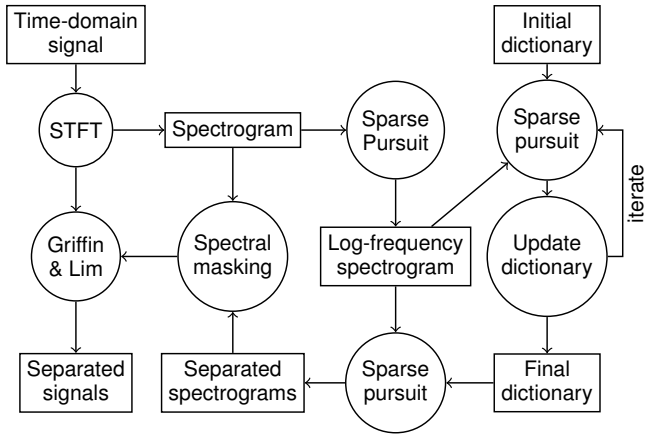


Figure 1. Data flow diagram for the proposed separation algorithm

The novelty of our approach lies in the *pitch-invariance* of the representation: In the dictionary, we save the relative amplitudes of the harmonics for each instrument, and we can use them to identify and synthesize the sounds of the instruments at any arbitrary pitch, without making assumptions about their tuning or range.

The sparse pursuit algorithm that we propose is designed to match a sampled spectrum with shifted non-negative continuous patterns. While it was developed with audio frequency spectra in mind, we foresee that it may have uses in other signal processing applications.

B. Related Work

A classical approach to the source separation problem is the use of non-negative matrix factorization (NMF) [3] in order to

Sören Schulze is with the working group Computational Data Analysis, Faculty 3, University of Bremen, Bibliothekstr. 5, 28359 Bremen, Germany (e-mail: sschulze@uni-bremen.de).

Emily J. King is with the Mathematics Department, Colorado State University, 1874 Campus Delivery, 111 Weber Bldg, Fort Collins, CO 80523, USA (e-mail: emily.king@colostate.edu).

obtain a dictionary. This was initially studied by Smaragdis and Brown [4] for the purpose of polyphonic music transcription and then applied to audio source separation by Wang and Plumbley [5]. An overview can be found in [6]. With these methods, the intended outcome is to have a new dictionary atom for every tone that is played by the instruments in the recording.

In many cases, a single musical instrument can generate different sounds which are perceptually similar and only vary in the pitch of the tones. Using the *constant-Q transform* (CQT) [7] as a log-frequency spectrogram, Fitzgerald et al. [8] generate a dictionary containing the frequency spectra of different instruments, which can be shifted on a fixed grid of semitones in order to apply them to different notes. This approach was later refined by Jaiswal et al. [9], [10], [11].

The advantage of this representation is that it can be applied to a large variety of musical instruments, as long as pitch-invariance is fulfilled. The drawback is that it requires the instruments to be tuned precisely to a known equal-temperament scale, which makes it impractical for real-world recordings with acoustic instruments.

By contrast, our representation supposes a certain structure of the harmonics that is satisfied by wind and string instruments. This reduces the number of variables to be learned for each instrument to the number of harmonics, and it enables it to adapt to changes in the width of the peaks and in inharmonicity (cf. [12]).

Beside the CQT, the *mel spectrogram* also has a logarithmic frequency axis, and there exists a time-frequency representation that combines favorable properties of both the CQT and the mel spectrogram [13]. However, the resolution of any spectrogram that was computed via classical means is eventually limited by the Heisenberg uncertainty principle (cf. [1], [14]).

C. The Musical Role of Sparsity

The representation of the time frames of a spectrogram of a music recording with a dictionary is in general not unique. If we consider wind and string instruments, their sound is dominated by a linear combination of sinusoids, which show up as horizontal lines in the spectrogram. Thus, there exists a trivial solution that assumes a single sinusoidal instrument which plays a large number of simultaneous tones. While this solution is valid, it is undesirable, as no separation is performed at all.

A similarly trivial solution is to construct different instruments for each time frame of the spectrogram. In this case, matching the constructed instruments with the actual instruments requires a lot of manual post-processing; also, instruments which play harmonically related notes may be mistaken for a single instrument, so this represention is again problematic for separation.

In order to attain meaningful solutions, it is essential to limit both the total number of instruments and the number of tones that are assumed to be played at the same time. The former is controlled by the layout of the dictionary, while the latter is a sparsity condition that requires the use of appropriate algorithms.

The constraints imposed by these numbers are supposed to encourage solutions that will appear meaningful to a human listener. Good results can be achieved if both numbers are known and sufficiently low, but blind separation meets its conceptual limits in case of very polyphonic works such as orchestral symphonies. One particularly difficult instrument is the pipe organ, where the combination of organs stops blurs the borders of what should be considered a single instrument (cf. [12], [15]).

D. Structure of this Paper

In Section II, we propose a novel general-purpose sparse pursuit algorithm that matches a sampled spectrum with non-negative continuous patterns. The algorithm is a modified version of orthogonal matching pursuit (OMP) [16] with a non-linear optimization step for refinement.

In Section III, we use this algorithm in order to convert an STFT magnitude spectrogram into a wideband pitch-invariant log-frequency spectrogram. In Section IV, we explain how we use the same algorithm (with slightly different parameter choices) and a dictionary representation of the harmonics in order to identify patterns of peaks related to the sounds of musical instruments in time frames of the spectrogram. Due to the non-linear optimization, we can identify the fundamental frequency, the width of the Gaussian, and the inharmonicity individually for each tone on a continuous scale.

In Section V, we expound the learning algorithm: For the dictionary update, we employ a modified version of Adam [17], which is a popular stochastic gradient descent algorithm that was initially developed for the training of deep neural networks. Our modifications adapt this algorithm to dictionary learning, preserving the relative scaling of certain components of the gradient and periodically resetting parts of the dictionary as needed.

In Section VI, we apply our algorithm on two mixtures that were recorded via acoustic instruments as well as on a set of computer-synthesized samples from the literature. We evaluate the performance of the overall algorithm via standard measures; further, we provide spectrograms for the separation result and use them to discuss the quality of the separation.

II. Sparse Pursuit Algorithm for Shifted Continuous Patterns

For both the transformation of the spectrogram and the identification of instruments inside the spectrogram, we need an algorithm to approximate a non-negative discrete mixture spectrum $Y[s] \geq 0$, $s \in \mathbb{Z}$, via shifted versions of continuous patterns $y_{\eta,\varpi}(s) \geq 0$, $s \in \mathbb{R}$, where $\eta \in \{0, \ldots, N_{\mathrm{pat}} - 1\}$ is a discrete index and $\varpi \in \mathbb{R}^{N_{\mathrm{par}}}$ is a set of continuous, real-valued parameters.

More specifically, we aim to identify amplitudes $a_j > 0$, shifts μ_j, indices η_j, and parameter sets ϖ_j such that:

$$Y[s] \approx \sum_j a_j y_{\eta_j,\varpi_j}(s - \mu_j),$$

for $s \in \mathbb{Z}$.

The first natural choice for a loss function is the ℓ_2 distance, but it is not ideal for use in magnitude frequency spectra, as it focuses very much on the high-volume parts of the spectrum, and the same applies to other ℓ_p (quasi-)distances for $p > 0$.

This problem is often approached by use of the β-divergence (cf. [18], [6]), which puts a high penalty on "unexplained" peaks in the spectrum. However, it is asymmetric, and while it is natural in NMF-based methods, it is difficult to integrate in the algorithm that we propose.

Instead, we remain with ℓ_2, but we *lift* low-volume parts of the spectrum via a concave power function:

$$R = \sum_s \left(Y[s]^q - \left(\sum_j a_j y_{\eta_j, \varpi_j}(s - \mu_j) \right)^q \right)^2,$$

with $q \in (0, 1]$.

Furthermore, we impose the *sparsity condition* that every value of η_j may only occur at most N_{spr} times in the linear combination.

Minimizing R is a highly non-convex and partly combinatorial problem, so we cannot hope to reach the perfect solution. Instead, we follow a *greedy* approach, using ideas from orthogonal matching pursuit (OMP) [16] and subspace pursuit [19].

We start with an empty index set $\mathcal{J}$ and then run the following steps in a loop:

1) Compute the cross-correlation between the residual

$$r[s] = Y[s]^q - \left(\sum_j a_j y_{\eta_j, \varpi_j}(s - \mu_j) \right)^q$$

(i.e., the lifted difference between the raw spectrum and the current reconstruction) and the sampled patterns. Assume a default parameter set ϖ_{nil}:

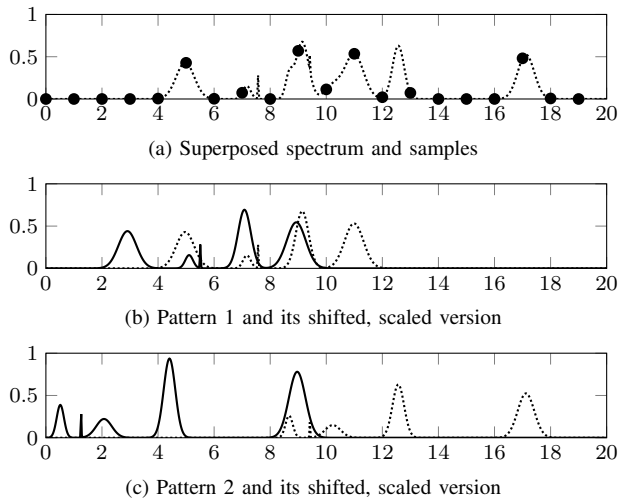
$$\rho[\mu, \eta] = \sum_i \frac{r[i] \left(y_{\eta, \varpi_{\mathrm{nil}}}(i - \mu) \right)^q}{\left\| y_{\eta, \varpi_{\mathrm{nil}}}[\cdot]^q \right\|_2}.$$

Preselect the N_{pre} combinations of (μ, η) with the greatest $\rho[\mu, \eta]$ and add them to the index set $\mathcal{J}$. For each preselected pair (μ_j, η_j), initialize $a_j = (\rho[\mu_j, \eta_j] / \| y_{\eta_j, \varpi_{\mathrm{nil}}}[\cdot]^q \|_2)^{1/q}$. Skip the combinations for which a_j is non-positive. If none are left, terminate.

2) Do non-linear optimization on a_j, μ_j, and ϖ_j, $j \in \mathcal{J}$, in order to minimize R, where $a_j \geq 0$ and $\varpi_j \in \Omega_\varpi$ with $\Omega_\varpi \subseteq \mathbb{R}^{N_{\mathrm{par}}}$.

3) For each $\eta = 0, \ldots, N_{\mathrm{pat}} - 1$, find the indices $j \in \mathcal{J}$ where $\eta_j = \eta$, and remove all but those with the N_{spr} highest amplitudes a_j such that, in the end, each pattern η is represented at most N_{spr} times in the index set $\mathcal{J}$. Re-run the non-linear optimization procedure on the now smaller index set $\mathcal{J}$.

4) If the loss R has decreased by less than the factor of $1 - \lambda$ compared to the previous iteration, with $\lambda \in (0, 1]$, restore all the values from the previous iteration and return them as the result.
Otherwise, if the count of iterations has reached N_{itr}, return the current parameters. If this is not the case, do another iteration.



(a) Superposed spectrum and samples

(b) Pattern 1 and its shifted, scaled version

(c) Pattern 2 and its shifted, scaled version

Figure 2. Example of the pursuit algorithm applied on a generic spectrum, which is a superposition of two patterns. The algorithm is only given the sampled points of the spectrum and the continuous patterns (solid lines). It finds the shifts and amplitudes for the patterns to reconstruct the original spectrum within numerical precision (dotted lines).

The hyperparameters N_{pat} and N_{spr} are usually determined by the application. For the q exponent, we usually pick $q = 1/2$, as this is the lowest value to keep R convex in a_j, which is beneficial to the optimization procedure. In some cases, better results can be achieved by choosing the value of q even lower, but this also increases the chance of divergence.

The hyperparameters λ and N_{itr} are safeguards to limit the runtime of the algorithm, such that the loop is not run indefinitely with marginal improvement in the non-linear optimization step. They also mitigate the problem of overfitting. The value of λ should be chosen slightly below 1; in practice, we find that $\lambda = 0.9$ yields good results. We limit the number of iterations to $N_{\mathrm{itr}} = 2 N_{\mathrm{spr}} N_{\mathrm{pat}}$, which is twice the overall sparsity level. The loop typically terminates due to insufficient decrease in R, not by exceeding N_{itr}.

As continuous functions are highly correlated with slightly shifted versions of themselves, we typically choose $N_{\mathrm{pre}} = 1$ in order to avoid the preselection of the same pattern multiple times for one feature in the spectrum.

The choice of the non-linear optimization algorithm is not critical, as long as it supports box bounds. We decided to employ the L-BFGS-B algorithm [20], [21], [22], which is fast even for high-dimensional problems.

In Figure 2, we applied the sparse pursuit algorithm to a superposition of two synthetic spectra. They are similar in shape to those obtained from audio signals but could also originate from other physical phenomena.

III. Computation of the Spectrogram

A spectrogram is a function defined on the time-frequency plane that is supposed to indicate to what extent a certain frequency is present in the recording at a given point in time.

The "canonical" time-frequency representation is the spectrogram obtained from the modulus of the STFT (cf. [1]), which

is defined via:

$$\mathcal{V}_w X(t,f) = \int_{-\infty}^{\infty} X(s)\, w(s-t)\, \mathrm{e}^{-i2\pi f s}\, \mathrm{d}s.$$

One particularly popular window with very favorable properties is the Gaussian:

$$w(t) = \exp\bigl(-t^2/(2\zeta^2)\bigr), \qquad \zeta \in \mathbb{R}.$$

For a sinusoidal signal $X(t) = \exp(i2\pi\nu t)$, this results in a horizontal line in the spectrogram:

$$\mathcal{V}_w X(t,f) = \mathcal{F}w(f-\nu)\, \mathrm{e}^{-i2\pi(f-\nu)t},$$

and

$$\mathcal{F}w(f-\nu) = a\, \exp(-(f-\nu)^2/(2\sigma^2))$$

with standard deviation $\sigma = 1/(2\pi\zeta)$ and amplitude $a = \sqrt{2\pi\zeta^2}$, where $\mathcal{F}$ is the unitary Fourier transform. In practice, we use an FFT-computed sampled version $Z[f,t] = |\mathcal{V}_w X(t/T, f/F)|$, where $T, F > 0$ are time and frequency units. While X has a sampling frequency of $f_\mathrm{s} = 48\,\mathrm{kHz}$, we subsample X by a factor of 256; thus, $T = 256/f_\mathrm{s} = 5.\overline{3}\,\mathrm{ms}$. Further, we set $\zeta = 1024/f_\mathrm{s}$ and cut w at $\pm 6\zeta$, yielding $F = f_\mathrm{s}/(12 \cdot 1024) = 3.90625\,\mathrm{Hz}$.

The problem is that the STFT spectrogram is not *pitch-invariant*: An octave equals a factor of 2 on the frequency axis, which is a different distance depending on the pitch of the tone.

In order to achieve pitch-invariance, one needs a representation with a logarithmic frequency axis. However, a naive transform of the modulus of the STFT would not only influence the position of the horizontal lines, but also their width. In order to overcome this problem, there exist two classical approaches:

- The mel spectrogram performs a logarithmic transform on the frequency axis of the STFT spectrogram and then applies smoothing along that axis in order to keep the widths consistent. This approach is limited by the Heisenberg uncertainty principle.
- The constant-Q transform is a discrete wavelet transform and can thus be understood as an STFT with differently dilated windows for each frequency. While it keeps the width of the horizontal lines constant on a logarithmic frequency axis, the time resolution will vary for different frequencies. This is problematic, as it results in simultaneously starting sinusoids first appearing in different time frames of the spectrogram.

As was shown by [23], the constant-Q transform can be turned into a mel spectrogram by applying additional smoothing along the time axis, but it is not possible to overcome the limitations of the Heisenberg uncertainty principle by classical means.

For narrow-banded signals, this is not a problem; the above methods can and have been used in order to provide a time-frequency representation for audio source separation. However, as we already have the method from Section II at hand, we may as well use it for spectrogram transformation and thus provide a method that works for wideband signals.

We thus set $Y = Z[\cdot, t]$ and assume a Gaussian

$$y_{0,\varpi}(s) = \exp\bigl(-s^2/(2F^2\sigma^2)\bigr), \qquad \varpi = (\sigma),$$

with $N_\mathrm{pat} = 1$ and $\varpi_\mathrm{nil} = (1/(2\pi\zeta))$.

Since the number of Gaussians in a spectrum can be high, we set $N_\mathrm{spr} = 1000$ to make sure they can all be represented. This makes the algorithm rather slow, so we choose $q = 1$ in order to make R more quadratic; as we aim to represent the spectrum with very low overall error, there is no need to lift certain features of the spectrum.

To reduce the number of iterations, we also set $N_\mathrm{pre} = 1000$. However, this comes with the aforementioned problem that the algorithm would select a lot of neighboring shifts. Thus, instead of computing the cross-correlation, we simply select the 1000 largest local maxima of the residual that satisfy $r[i] \geq r[i+k]$ for $|k| \leq 3$ and assume their height as amplitude.

To allow for high-detail representation, we set $\lambda = 1$. The maximum number of iterations is $N_\mathrm{itr} = 20$, but the algorithm often terminates before that.

After having identified the Gaussian peaks in the sampled STFT magnitude spectrogram $Z[f,t]$, we resynthesize them in another magnitude spectrogram $U[\alpha, t]$, applying a logarithmic frequency transform $\alpha(f) = \alpha_0 \log_2(f/f_0)$ to the mean frequencies μ_j, $j \in \mathcal{J}$. With $f_0 = 20\,\mathrm{Hz}/f_\mathrm{s} \cdot 12 \cdot 1024 = 5.12$ and $\alpha_0 = 1024/10 = 102.4$, we can, assuming a sampling frequency of $f_\mathrm{s} = 48\,\mathrm{kHz}$ and $\alpha = \{0, \ldots, 1023\}$, represent 10 octaves from 20 Hz to 20.48 kHz.

The algorithm can also be used without modification for compact disc (CD) recordings with a sampling frequency of $f_\mathrm{s} = 44.1\,\mathrm{kHz}$. In this case, the represented audio frequency range is from 18.375 Hz to 18.816 kHz.

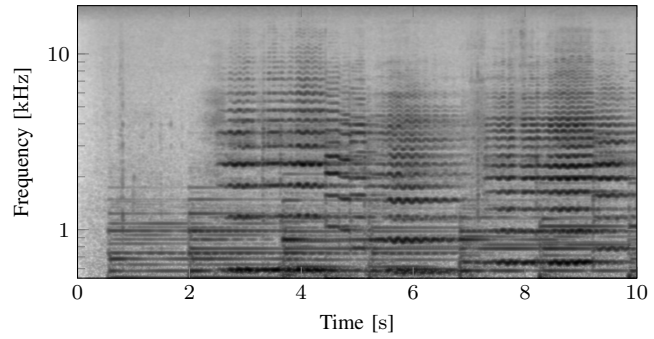
For Figure 3, we performed different transforms on an excerpt of a commercial recording of a piece for violin and piano. The mel spectrogram in Figure 3a had to be cut off at 530 Hz in order to maintain a constant time-log-frequency resolution. The constant-Q transform in Figure 3b can represent lower frequencies, but its time-log-frequency resolution varies with frequency: Clearly, the tones with lower frequencies have a wider time spread in the representation than those with higher frequencies, giving an inconsistent image in the individual time frames.

Our proposed sparsity-based transform in Figure 3c does not have this problem: It aligns the tones properly along the time axis like the mel spectrogram, but it can represent much lower frequencies.

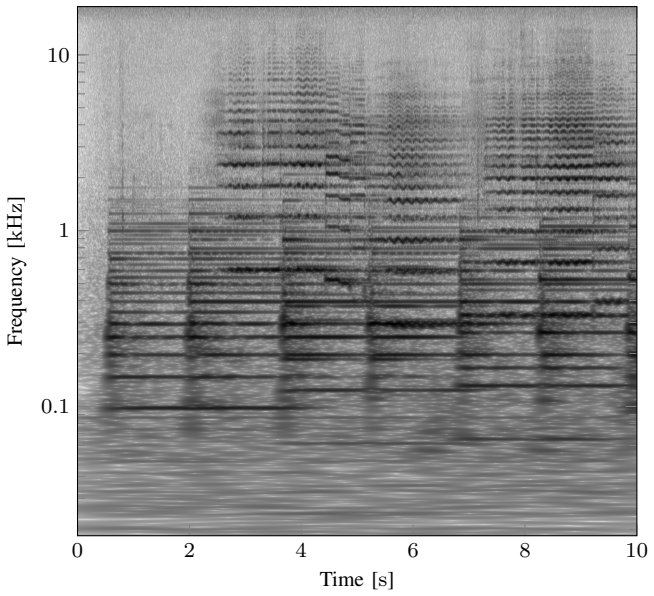
As our proposed representation is specifically designed for sinusoids, it largely fails to represent other sounds; in this case, however, this is even beneficial, as it removes portions of the spectrogram that do not correspond to the tones that we aim to represent (creating the white regions in Figure 3c). From this perspective, we can say that it *denoises* the spectrogram.[1]

However, it should be kept in mind that the uncertainty principle cannot be "tricked" arbitrarily; if two sinusoids have very low and very similar frequencies, their representations in the STFT spectrogram will overlap greatly, and our algorithm may fail to tell them apart. On the other hand, if a peak is slightly perturbed, it may also occur that the algorithm will identify one single sinusoid as two.

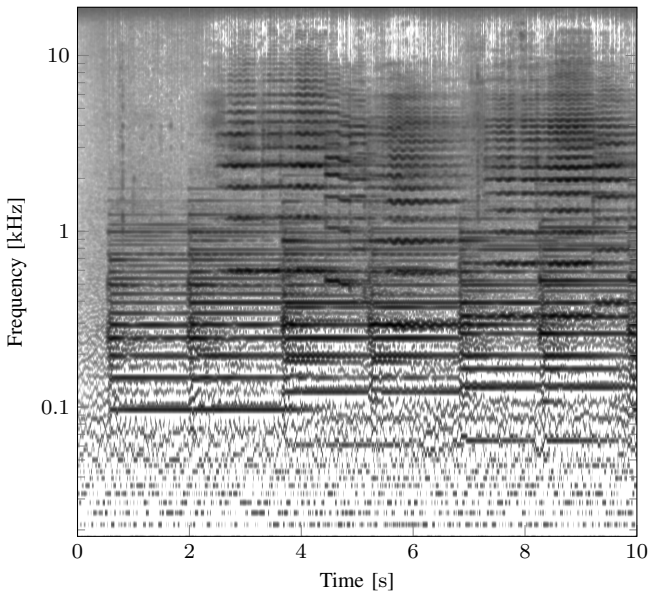
[1]To our separation algorithm, anything non-sinusoidal is noise. This does not imply, however, that these parts of the signal are undesirable for a human listener.



(a) Mel spectrogram of the recording



(b) Constant-Q transform of the recording



(c) Sparsity-based representation of the recording

Figure 3. Log-frequency spectrograms of the beginning of the 1st mvt. of the sonata no. 1 for violin and piano by Johannes Brahms (op. 100). The grayscale axis is logarithmic and normalized to a dynamic range of 100 dB for each plot. Performance by Itzhak Perlman and Vladimir Ashkenazy. Remastered CD recording by EMI Classics, 1999.

Some parts of the noise do get mistaken for sinusoids and are thus carried over to the log-frequency spectrogram. In the low frequencies, this creates the illusion of sparsity in the log-frequency spectrogram, as can be seen in Figure 3c. The positions of those horizontal lines correspond to the transformed frequencies of the pixels in the linear-frequency spectrogram.

IV. Model Representation of the Spectrogram

We aim to represent the discrete log-frequency spectrogram $U[\alpha, t]$, $\alpha, t \in \mathbb{Z}$, which contains the superposed sound of the musical instruments, via the computationally isolated sound of the instruments, without knowing their exact characteristics beforehand.

A simple model for many musical instruments (particularly string and wind instruments) is the wave equation, which has sinusoidal solutions (the *harmonics*) at frequencies $f_h = hf_1^\circ$, $h = 1, \dots, N_{\mathrm{har}}$, where $f_1^\circ > 0$ is the *fundamental frequency*. However, many string instruments (especially the piano in its high notes) have non-negligible stiffness in their strings, leading to a fourth-order equation which has solutions $f_h = (1 + bh^2)^{1/2} hf_1^\circ$, with the inharmonicity parameter $b \geq 0$ (cf. [12]).

In order to avoid negative frequencies in the spectrogram, we model our time-domain signal for the jth tone as a linear combination of complex exponentials:

$$x_j(t) = \frac{1}{\sqrt{2\pi\sigma^2}} \sum_{h=1}^{N_{\mathrm{har}}} a_{j,h} \cdot \mathrm{e}^{i2\pi(f_{j,h}t + \varphi_{j,h})},$$

with phase values $\varphi_{j,h} \in [0, 2\pi)$.

We assume that the images of these sinusoids superpose linearly in the spectrograms. In reality, this is not the case in the presence of non-constructive interference (*beats*), but if we accept the error introduced by this common simplification, we can approximate $Z[f, t]$ via:

$$z[f, t] \coloneqq \sum_{j,h} a_{j,h,t} \cdot \exp\left(-\frac{\big(f - f_{j,h,t}\big)^2}{2F^2\sigma_{j,t}^2} \right),$$

where $a_{j,h,t}$ is the amplitude of the hth harmonic of the jth tone in the tth time frame, and $f_{j,h,t}$ is the respective frequency. For the log-frequency spectrogram $U[\alpha, t]$, this transforms to the following approximation:

$$u[\alpha, t] \coloneqq \sum_{j,h} a_{j,h,t} \cdot \exp\left(-\frac{\big(\alpha - \alpha_{j,h,t}\big)^2}{2F^2\sigma_{j,t}^2} \right),$$

with $\alpha_{j,h,t} = \alpha(f_{j,h,t}) = \alpha((1 + b_{j,t}h^2)^{1/2}h) + \alpha(f_{j,1,t}^\circ)$.

We further make the simplifying assumption that the sound of a musical instrument is constant over the duration of a tone, and that the relation of the amplitudes of the harmonics is constant with respect to pitch and volume. In practice, this introduces a certain error, but we will later take measures in order to mitigate it. We thus save the relative amplitudes of the instruments in a *dictionary*, which is a two-dimensional data structure $D \in [0, 1]^{N_{\mathrm{har}} \times N_{\mathrm{pat}}}$. Introducing an overall amplitude $a_{j,t}$ for each tone, we can express $a_{j,h,t} = D[h, \eta_{j,t}]\, a_{j,t}$, where $\eta_{j,t}$ is the instrument by which the tone is played.

Our pursuit algorithm can now be applied by setting:

$$y_{\eta_{j,t},\varpi_{j,t}}(\alpha) = \sum_h D[h,\eta_{j,t}]\cdot \exp\left(-\frac{\big(\alpha-\alpha((1+b_{j,t}h^2)^{1/2}h)\big)^2}{2F^2\sigma_{j,t}^2}\right),$$

with $\varpi_{j,t} = (\sigma_{j,t}, b_{j,t})$ and $\mu_{j,t} = \alpha(f^\circ_{j,1,t})$. The initial value is $\varpi_{\mathrm{nil}} = (1/(2\pi\zeta), 0)$.

V. Dictionary Learning

A. Scheme

In order to train the dictionary, we pursue a stochastic alternating-optimization approach. First the dictionary is initialized; for each $\eta = 0, \ldots, N_{\mathrm{pat}} - 1$, we generate a uniformly distributed random vector $d \in [0,1)^{N_{\mathrm{har}}}$ and an exponent e that is Pareto-distributed with a scale parameter of 2 (to make sure that $e \geq 1$, guaranteeing a minimum decay rate), and we set $D[h,\eta] = d[h]/h^e$.

Given an initial dictionary, a random time frame $U[\cdot,t]$ of the log-frequency spectrogram of the recording is chosen, and a sparse dictionary representation is computed. Afterwards, the dictionary is updated to improve the representation, and the process is repeated. Finally, after the learning process, the dictionary is applied for separation of the entire sample.

We set the number of patterns to be generated from the dictionary to twice the expected number of instruments in the recording ($N_{\mathrm{pat}} = 2N_{\mathrm{ins}}$), allowing for some redundancy during the training.

B. Dictionary Update

Classically, dictionary learning is performed via techniques like NMF [3], [18], K-SVD [24], or tensor factorization. However, the first two methods do not account for our pitch-invariant structure of the data. The latter does, but only for a fixed number of frequency shifts. Moreover, all of these methods become slow when the amount of data is large.

While the use of stochastic gradient descent for dictionary learning has been common for many years (cf., e.g., [25]), new methods have been arising very recently due to their applications in deep learning. One of the most popular methods for this purpose is Adam [17]. Its underlying idea is to treat the gradient as a random variable and, for each component, compute unbiased estimates $\hat{v}_1, \hat{v}_2$ for the first and second moments, and choose the step size proportional to $\hat{v}_1/\sqrt{\hat{v}_2}$. If the derivative of the ith component is constant, then $\hat{v}_1[i]/\sqrt{\hat{v}_2[i]} = \pm 1$, in which case a large step size can be used. If the derivative oscillates a lot, however, then $\hat{v}_1[i]/\sqrt{\hat{v}_2[i]}$ will also be small and thereby dampen the oscillation in that direction.

The standard formulation of Adam is completely independent of the scale of the derivatives. This makes it easy to control the absolute step size of the components, but it destroys the Landweber regularization property of gradient descent, which automatically decreases the step size for components whose partial derivative is small, taking into account the scaling of different harmonics.

Our first modification to Adam is that while we still estimate the first moments for each dictionary entry (i.e., for each instrument and for each harmonic), we only compute one second moment estimate for each instrument, which is the arithmetic mean over the all the estimates for the harmonics. With this, we restore the regularization property and prevent excessive change of the components that have small values.

Furthermore, we require all entries in the dictionary to be non-negative, since negative harmonic amplitudes would be unphysical. For consistency, we also require that no entries be larger than 1, so we end up with the box constraint that $D[h,\eta] \in [0,1]$ for $h = 1, \ldots, N_{\mathrm{har}}$, $\eta = 0, \ldots, N_{\mathrm{pat}} - 1$. To enforce this, we project each component to $[0,1]$ after the end of a step.

Finally, we have to tackle the problem that due to the stochastic nature of the optimization procedure, dictionary entries for a particular supposed instrument may diverge to a point where they will not be used by the identification algorithm any more and thus not contribute to the separation. For this purpose, we track the sum of the amplitudes associated with a specific instrument in the past. In regular intervals, we sort the instruments in the dictionary by the ratio of the amplitude sum versus the number of iterations since its initialization (minus a small head start that benefits new instrument entries); then, we prune the dictionary by reinitializing the entries for those supposed instruments where the ratio is lowest, leaving the N_{ins} instruments with the highest ratios intact.

C. Separation and Resynthesis

After the dictionary has been trained by alternating between identification and dictionary update, we represent the entire recording by running the identification/pursuit algorithm on each time frame $U[\cdot,t]$ for $t = 0, \ldots, n-1$ (where n is the number of time frames in the spectrogram) with those N_{ins} instruments in the dictionary that were left intact after the latest renewal. This time, however, we need a linear-frequency spectrogram, so we apply the reverse transformation $f(\alpha) = f_0\, 2^{\alpha/\alpha_0}$ on the means of the Gaussians and reconstruct the spectrogram for the ηth instrument via:

$$z_\eta[f,t] \coloneqq \sum_{\substack{j,h \\ \eta_{j,t}=\eta}} a_{j,h,t} \cdot \exp\left(-\frac{\big(f - f_{j,h,t}\big)^2}{2F^2\sigma_{j,t}^2}\right).$$

For the generation of the time-domain signal, we use the classical algorithm by Griffin and Lim [2], which iteratively approximates the signal whose corresponding STFT magnitude spectrogram is (in the ℓ_2 sense) closest to the given one. As initial value, we give the phase of the STFT of the original signal.

While more sophisticated phase retrieval methods have been developed recently (e.g., [26]), the algorithm by Griffin and Lim is well-established, robust, and simple.

D. Spectral Masking

As an optional post-processing step, we can mask the spectrograms from the dictionary representation with the

spectrogram from the original recording. This method was proposed in [9], [10]:

$$\tilde{z}_\eta[f,t] \coloneqq \frac{z_\eta[f,t]}{z[f,t]} \cdot Z[f,t].$$

In practice, a tiny value is added to the denominator in order to avoid division by zero.

With this procedure, we make sure that the output spectrograms do not have any artifacts at frequencies that are not present in the original recording. Another benefit is mentioned in [9]: In cases where the sound of an instrument is not perfectly invariant with respect to pitch and volume, the masking can correct this.

A potential drawback with masking is that destructive interference in the original spectrogram may alter the spectrograms of the isolated instruments.

VI. Experimental Evaluation

We generate the log-frequency spectrogram as specified in Section III. For the dictionary, we use $N_{\mathrm{har}} = 25$ harmonics.

A. Performance Measures

Vincent et al. [27] define the signal-to-distortion ratio (SDR), the signal-to-interference ratio (SIR), and the signal-to-artifacts ratio (SAR). These measures have become the de facto standard for the performance evaluation of blind audio source separation. Assuming original signals $(X_\eta)_{\eta=0,\dots,N_{\mathrm{ins}}-1}$ and reconstructed signals $(x_\eta)_{\eta=0,\dots,N_{\mathrm{ins}}-1}$, those quantities (to be interpreted in dB) are defined as:

$$\mathrm{SDR}_\eta = 10 \cdot \log_{10} \frac{\left\|\mathcal{P}_{X_\eta}(x_\eta)\right\|_2^2}{\left\|\mathcal{P}_{X_\eta}(x_\eta) - x_\eta\right\|_2^2},$$

$$\mathrm{SIR}_\eta = 10 \cdot \log_{10} \frac{\left\|\mathcal{P}_{X_\eta}(x_\eta)\right\|_2^2}{\left\|\mathcal{P}_{X_\eta}(x_\eta) - \mathcal{P}_X(x_\eta)\right\|_2^2},$$

$$\mathrm{SAR}_\eta = 10 \cdot \log_{10} \frac{\left\|\mathcal{P}_X(x_\eta)\right\|_2^2}{\left\|\mathcal{P}_X(x_\eta) - x_\eta\right\|_2^2},$$

where $\mathcal{P}_{X_\eta}$ is the orthogonal projection on X_η, while $\mathcal{P}_X$ is the orthogonal projection on $\mathrm{span}\{X_\eta : \eta = 0, \dots, N_{\mathrm{ins}} - 1\}$.

The SDR is an "overall" performance measure that incorporates all kinds of errors in the reconstructed signal; the orthogonal projection in the numerator leads to a value of $-\infty$ if the original signal and the reconstructed signal are uncorrelated. The SIR is similar, but it ignores any artifacts that are uncorrelated with the original signals. The SAR only measures the artifacts and ignores interference; it is constant with respect to permutations of the original signals. Due to the projection, those measures are independent of the scale of the reconstruction, but they are very sensitive to phase mismatch: The projection of a sinusoid on its $90°$-shifted copy will be zero, even though the signals are otherwise identical.

In order to find the right mapping between the synthesized and the original signals, we permute the synthesized signals such that the mean SIR over all instruments is highest.

B. Separation of Recorder and Violin Sounds

We used the 8th piece from the 12 Basset Horn Duos by Wolfgang A. Mozart (KV 487) in an arrangement by Alberto Gomez Gomez for two recorders[2]. The upper part was played on a soprano recorder, and the lower part was played on a violin. These instruments are easily distinguishable, as the recorder has an almost sinusoidal sound, while the sound of the violin is sawtooth-like, with strong harmonics [12].

The instrument tracks were recorded separately, while a metronome/"play-along" track was provided via headphones. Evenness of the tone was favored over musical expression. We combined the tracks by adding the two digital signals with no post-processing other than adjustment of volume and overall timing and let the algorithm run with $N_{\mathrm{itr}} = 100000$ training iterations[3], with $N_{\mathrm{ins}} = 2$ and $N_{\mathrm{spr}} = 1$.

This procedure was performed with random seeds $0, \dots, 9$; the results are displayed in Table I. As can be expected, the recorder is universally better represented than the violin, and spectral masking leads to considerable improvements in both domains especially for the violin. This complies with the explanation in [9] that spectral masking helps represent instruments with more diverse spectra, as the violin has 4 different strings, and its sound is very sensitive to technique.

Table I
Performance measures for the evaluation of the separation of recorder and violin, once without and once with spectral masking. Indicated are the best-case numbers (maximum mean SDR) as well as the means and sample standard deviations over 10 runs.

Mask	Instrument	Measure	Best	Mean	Stdev
No	Recorder	SDR	12.8	11.7	1.4
		SIR	31.8	31.5	1.9
		SAR	12.9	11.8	1.4
	Violin	SDR	7.1	6.3	1.4
		SIR	24.8	23.2	2.5
		SAR	7.2	6.4	1.4
Yes	Recorder	SDR	15.3	13.9	2.0
		SIR	31.9	31.6	1.9
		SAR	15.4	14.0	2.0
	Violin	SDR	12.0	10.6	2.1
		SIR	24.3	21.9	3.3
		SAR	12.3	11.0	2.0

For phase reconstruction, we used merely one iteration (i.e., only one magnitude adjustment and one projection) of the Griffin-Lim algorithm in order to preserve the phase of the original spectrogram as much as possible.

The aural impression of the results with different random seeds is largely very similar. In terms of mean SDR, the random seed of 7 yields the best results. While some artifacts and interference are audible, the generated audio data provides a good aural representation of the actually played tracks. The

[2] https://imslp.org/wiki/12_Horn_Duos,_K.487/496a_(Mozart,_Wolfgang_Amadeus)

[3] We already achieve similarly good performance with $N_{\mathrm{itr}} = 10000$ iterations, but more iterations make the result more consistent with respect to initial values.

only tone[4] that is misidentified over a long period of time is a recorder tone that interferes with the even-numbered harmonics of the violin tone that is played at the same time and is one octave lower. In this case, the third harmonic of the violin tone is erroneously identified as the recorder tone.

Spectrograms of the original recording and the synthesized representations are displayed in Figure 4. The original spectrogram contains broad-spectrum components ("noise") that do not fit the dictionary model and thus cannot be represented, so they are not found in the output spectrograms. The choice of $N_{\mathrm{har}} = 25$ must be regarded as a compromise: Although the sound of the violin could be represented more accurately with an even higher numbers of harmonics, this would increase both the computation time of the algorithm and also the number of variables to be trained. The incorrectly identified recorder tone corresponds to the rightmost set of horizontal lines in Figure 4b. It is not audible when the synthesized audio files are mixed back together.

Since spectral masking is only applied on the linear-frequency spectrograms, its effects cannot be seen in Figure 4.

C. Separation of Clarinet and Piano Sounds

We recorded the same piece on clarinet and piano. Compared to the combination of recorder and violin, it was much more difficult to obtain consistently good results. We thus ran the algorithm for $N_{\mathrm{itr}} = 10000$ iterations with different random initial values. For a random seed value of 5, we obtained the values presented in Table II.

Table II
PERFORMANCE MEASURES FOR THE BEST-CASE RUN OF THE SEPARATION OF CLARINET AND PIANO, WITH SPECTRAL MASKING.

Instrument	SDR	SIR	SAR
Clarinet	4.1	18.6	4.3
Piano	1.5	9.6	2.7

As it turns out, the onset (*attack*) of a piano tone is especially problematic to represent in our dictionary model, as it exhibits different spectral characteristics than the rest of the tone.

D. Comparison

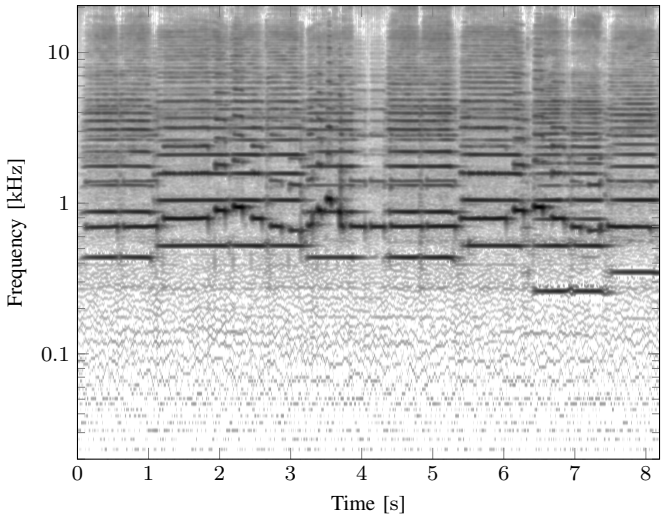
We ran our algorithm on the data that was used in [9], [10], [11], which consists of computer-synthesized samples with two instruments, each playing one tone at a time. Again, we set $N_{\mathrm{itr}} = 10000$ and selected the best result (in terms of mean SDR) out of 10 runs (with random seeds $0, \ldots, 9$) for each sample. No further adjustments were conducted. The performance measures are displayed in Figure 5 and Table III.

It can be seen that for certain samples, our algorithm performs very well, while for others, it fails to produce acceptable results. When comparing the means, our algorithm is slightly inferior to [9], [10], [11].[5]

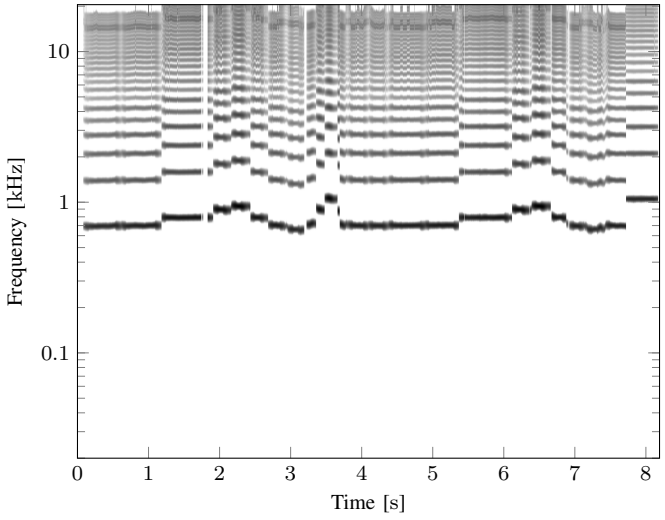
Our explanation for this is that our algorithm assumes much looser constraints on the data that it gets, as it accepts arbitrary

[4]which occurs 4 times in total, due to repetitions of the passage

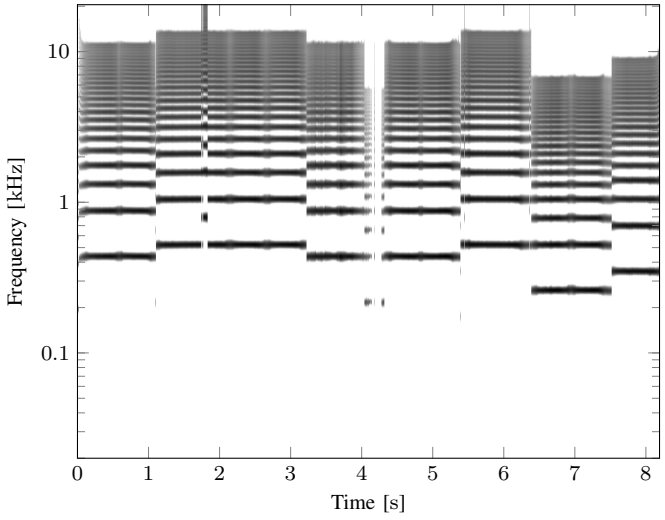
[5]We could not compare the performance on the individual samples, as those numbers are not available to us.



(a) Original, generated via the sparse pursuit method



(b) Synthesized recorder track



(c) Synthesized violin track

Figure 4. Log-frequency spectrograms for beginning of the recorded piece and the synthesized tracks with a random seed of 7. The grayscale axis is logarithmic and normalized to a dynamic range of 100 dB for each plot.

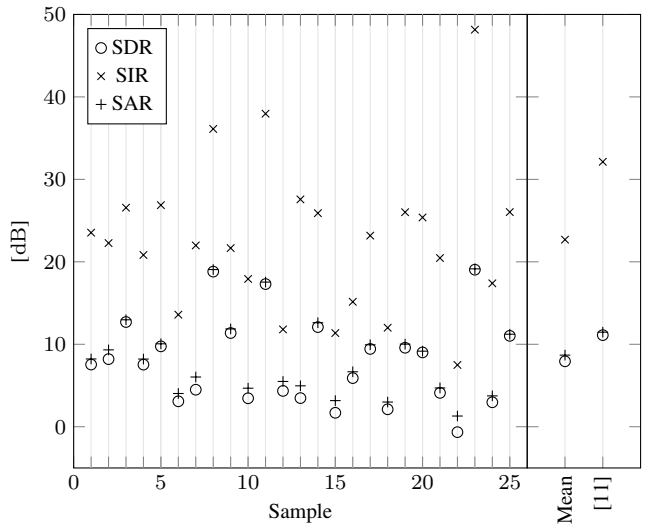


Figure 5. Performance of our algorithm applied on the audio samples from [9], [10], [11] (best-case run out of 10 for each sample). The means over the samples with our algorithm are compared to the mean values given in [11].

Table III
COMPARISON OF OUR ALGORITHM TO [9], [10], [11] ON THE DATA USED THEREIN (MEANS OVER ALL INSTRUMENTS AND ALL SAMPLES IN THE BEST CASES).

Algorithm	SDR	SIR	SAR
[9]	8.9	23.7	9.7
[10]	10.9	25.4	11.5
[11]	11.1	32.1	11.5
Ours	7.9	22.7	8.7

tones in the audible range. By constrast, in [9], [10], [11], the expected fundamental frequencies for the instruments are hardcoded in the algorithm due to prior knowledge. In [9], 7 values are allowed per sample, while in [10], this number was invidually adjusted to 4–9 values for each sample in order to achieve maximum performance figures; in [11], those were 5–12 values. Further, the algorithms can exploit the fact that the tone ranges for the respective instruments in the samples were chosen to have little or no overlap. In the case of no overlap, such distinctive information would even make it possible to separate instruments with identical frequency spectra, but this would violate our notion of blind separation.

As can be seen in Table III, the individual adjustments that were conducted in [10] had a much greater effect on the performance than the algorithmic improvements in [11].

Applying the algorithms in [9], [10], [11] to our data would not be possible, as those algorithms require, due to their data representation, perfectly consistent equal temperament tuning, which wind instruments and string instruments without frets do not satisfy.

We conclude that the out-of-the-box performance of our algorithm is in means slightly inferior to the figures in [9], [10], [11] on the samples used therein, but this is compensated by its vastly greater flexibility, which enables it to operate on real-world acoustic signals and eliminates the need for prior specification of the tuning or range of the instruments.

VII. Conclusion and Future Work

We developed a novel algorithm to represent discrete mixture spectra as sparse shifted linear combinations of analytically given non-negative continuous patterns. We applied this algorithm to spectrograms of audio recordings, first to convert an STFT magnitude spectrogram into a log-frequency spectrogram, then to identify patterns of peaks related to the sounds of musical instruments in the context of a dictionary learning algorithm based on Adam, a method that originates from the field of deep learning.

This led us to an algorithm to perform blind source separation on polyphonic music recordings of wind and string instruments that is unique in its ability to represent and process realistic signals from actual music recordings without making any assumption about the tuning of the instruments, while maintaining performance that was previously achieved only on synthetic data.

The algorithm does not require any prior information other than the number of instruments and an upper bound for the sparsity level. It is especially suitable for instruments where the exact frequency of a tone is not fixed by the construction of the instrument but at least partly depends on playing technique.

We note, however, that blind source separation always requires favorable data: Right now, our algorithm can only cope with music played on string and/or wind instruments; the spectra of the individual instruments must be clearly distinguishable by the amplitudes of the harmonics, they must be sufficiently pitch- and volume-invariant with only little overall variation in timbre, and the sparsity level must be rather strict.

While our representation of the sounds of musical instruments appears to be well-fitting for the instruments that we consider, we believe that the complexity of the problem brings a classical alternating optimization approach to its limits. We thus intend to experiment with a hybrid approach where some parts of the algorithm are replaced with a neural network.

In our application, the frequency shift of the spectrum is caused by a change in pitch. Another common source for frequency shifts in various areas (such as communication technology or astronomy) is the Doppler effect. We believe that this could open new applications for our pursuit algorithm and potentially also the dictionary learning algorithm.

VIII. Resources

The software is presented along with audio samples and the original input data on the institute website[6]. The source code is available on GitHub[7].

In the supplementary material, we provide pseudo-code with a description of the implementation details as well as some non-essential figures with commentary.

[6]https://www.math.colostate.edu/~king/software.html#Musisep
[7]https://github.com/rgcda/Musisep

ACKNOWLEDGEMENTS

The first author acknowledges funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Projektnummer 281474342/GRK2224/1.

The authors would like to thank Bernhard G. Bodmann, Gitta Kutyniok, and Monika Dörfler for engaging discussions on the subject, and Kara Tober for playing the clarinet samples.

REFERENCES

[1] K. Gröchenig, *Foundations of Time-Frequency Analysis*. Springer, 2001.
[2] D. Griffin and J. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 2, pp. 236–243, 1984.
[3] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.
[4] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on*. IEEE, 2003, pp. 177–180.
[5] B. Wang and M. D. Plumbley, "Musical audio stream separation by non-negative matrix factorization," in *Proc. UK Digital Music Research Network (DMRN) Summer Conf.*, 2005.
[6] C. Févotte, E. Vincent, and A. Ozerov, "Single-channel audio source separation with NMF: divergences, constraints and algorithms," in *Audio Source Separation*. Springer, 2018, pp. 1–24.
[7] J. C. Brown, "Calculation of a constant Q spectral transform," *J. Acoust. Soc. Am.*, vol. 89, no. 1, pp. 425–434, 1991.
[8] D. Fitzgerald, M. Cranitch, and E. Coyle, "Shifted non-negative matrix factorisation for sound source separation," in *Statistical Signal Processing, 2005 IEEE/SP 13th Workshop on*. IEEE, 2005, pp. 1132–1137.
[9] R. Jaiswal, D. FitzGerald, D. Barry, E. Coyle, and S. Rickard, "Clustering NMF basis functions using shifted NMF for monaural sound source separation," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 245–248.
[10] R. Jaiswal, D. Fitzgerald, E. Coyle, and S. Rickard, "Shifted NMF using an efficient constant-Q transform for monaural sound source separation," in *22nd IET Irish Sig. and Sys. Conf.* IET, 2011.
[11] ——, "Towards shifted NMF for improved monaural separation," in *24th IET Irish Signals and Systems Conference*. IET, 2013.
[12] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*. Springer Science & Business Media, 2012.
[13] S. Schulze and E. J. King, "A frequency-uniform and pitch-invariant time-frequency representation," *PAMM*, 2019, to appear.
[14] G. B. Folland and A. Sitaram, "The uncertainty principle: A mathematical survey," *J. Fourier Anal. Appl.*, vol. 3, no. 3, pp. 207–238, 1997.
[15] W. H. Barnes, *The Contemporary American Organ*, 8th ed. J. Fischer and Bro., 1964.
[16] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.
[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[18] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Adv. Neural Inf. Process. Syst.*, 2001, pp. 556–562.
[19] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.
[20] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM J. Sci. Comput.*, vol. 16, no. 5, pp. 1190–1208, 1995.
[21] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Trans. Math. Softw.*, vol. 23, no. 4, pp. 550–560, 1997.
[22] J. L. Morales and J. Nocedal, "Remark on "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization"," *ACM Trans. Math. Softw.*, vol. 38, no. 1, p. 7, 2011.
[23] M. Dörfler, T. Grill, R. Bammer, and A. Flexer, "Basic filters for convolutional neural networks: Training or design?" *Neural Comput. Appl.*, pp. 1–14, 2018.
[24] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, 2006.
[25] M. Aharon and M. Elad, "Sparse and redundant modeling of image content using an image-signature-dictionary," *SIAM J. Imaging Sci.*, vol. 1, no. 3, pp. 228–247, 2008.
[26] G. E. Pfander and P. Salanevich, "Robust phase retrieval algorithm for time-frequency structured measurements," *SIAM J. Imaging Sciences*, vol. 12, no. 2, pp. 736–761, 2019.
[27] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Trans. Audio, Speech, Language Process.*, vol. 14, no. 4, pp. 1462–1469, 2006.

Sören Schulze received his B.Sc. in 2013 and his M.Sc. in 2016, both in industrial mathematics at the University of Bremen. For his master thesis, in which he began dealing with the instrument separation problem, he earned the department prize. He is curently working as a junior researcher in the Computational Data Analysis group in Bremen as a member of the DFG Research Training Group 2224. His research interests include signal and audio processing, machine learning, numerical mathematics, and applied harmonic analysis.

Emily J. King Emily J. King earned a B.S. in Applied Mathematics and an M.S. in Mathematics from Texas A&M University, College Station, Texas, U.S.A., in 2003 and 2005, respectively. She then received a Ph.D. in Mathematics from the University of Maryland, College Park, Maryland, U.S.A., in 2009. From 2009 to 2011, she was an IRTA Postdoctoral Fellow in the Laboratory for Integrative and Medical Biophysics at the National Institutes of Health in Bethesda, Maryland, U.S.A., and simultaneously a Postdoctoral Research Associate with the Norbert Wiener Center at the University of Maryland. As an Alexander von Humboldt Postdoctoral Fellow, she worked from 2011 to 2013 at the University of Osnabrück, the University of Bonn, and the Technical University of Berlin, all in Germany. While in Berlin, she also was a Postdoctoral Faculty Member at the Berlin Mathematical School. From 2013 to 2019, she was employed at the University of Bremen, Bremen, Germany, first as a Postdoctoral Researcher and then as a Juniorprofessor leading the research group Computational Data Analysis. Since August 2019, she is an Assistant Professor at Colorado State University, doing research in the fields of algebraic and applied harmonic analysis, signal and image processing, data analysis, and frame theory.

Photo credit: Uni Bremen / Kai Uwe Bohn.

PSEUDO-CODE

We will now present the mentioned algorithms in more detail via pseudo-code. First, Figure 6 describes the sparse pursuit/identification algorithm. It takes as arguments the dictionary, the sample vector, a selector function, the sparsity levels, and the sum of the previous amplitudes for each pattern (which will become important for dictionary renewal). In the non-linear optimization step, it calls the L-BFGS-B minimizer to minimize the loss R with respect to the given parameters.

function pursuit($D, Y,$ select, $N_{\mathrm{pre}}, N_{\mathrm{spr}}, A$)
  $\mathcal{J} \leftarrow \emptyset$
  $r \leftarrow Y^q$
  loop N_{itr} times
    $a_j \leftarrow 0,\ \varpi_j \leftarrow \varpi_{\mathrm{nil}}$
      for $j \in \{1, \dots, N_{\mathrm{spr}} + N_{\mathrm{pre}}\} \setminus \mathcal{J}$
    $\mathcal{J}_{\mathrm{new}} \leftarrow \mathrm{sort}(\{1, \dots, N_{\mathrm{spr}}+N_{\mathrm{pre}}\}\setminus\mathcal{J})[1, \dots, N_{\mathrm{pre}}]$
    $a_{\mathcal{J}_{\mathrm{new}}}, \mu_{\mathcal{J}_{\mathrm{new}}}, \eta_{\mathcal{J}_{\mathrm{new}}}$
      $\leftarrow$ select$(r, y_0, \dots, y_{N_{\mathrm{pat}}-1}, \mathcal{J}_{\mathrm{new}}, N_{\mathrm{pre}})$
    $\mathcal{J} \leftarrow \mathcal{J} \cup \mathcal{J}_{\mathrm{new}}$
    $a_{\mathcal{J}}, \mu_{\mathcal{J}}, \varpi_{\mathcal{J}} \leftarrow \mathrm{bfgs}(R, D, Y, a_{\mathcal{J}}, \mu_{\mathcal{J}}, \varpi_{\mathcal{J}}, \eta_{\mathcal{J}})$
      $a_{\mathcal{J}} \geq 0,\ \varpi_{\mathcal{J}} \in \Omega_{\varpi}$
    for $\eta = 0, \dots, N_{\mathrm{pat}} - 1$ **do**
      $\mathcal{J}_\eta \leftarrow \big(\arg\mathrm{sort}_{j \in \{1,\dots,N_{\mathrm{spr}}+N_{\mathrm{pre}}\},\, \eta_j = \eta} a_j\big)$
        $[1, \dots, N_{\mathrm{spr}}]$
    $\mathcal{J} \leftarrow \bigcup_{\eta \in \{0,\dots,N_{\mathrm{pat}}-1\}} \mathcal{J}_\eta$
    $a_{\mathcal{J}}, \mu_{\mathcal{J}}, \varpi_{\mathcal{J}} \leftarrow \mathrm{bfgs}(R, D, Y, a_{\mathcal{J}}, \mu_{\mathcal{J}}, \varpi_{\mathcal{J}}, \eta_{\mathcal{J}})$
      $a_{\mathcal{J}} \geq 0,\ \varpi_{\mathcal{J}} \in \Omega_{\varpi}$
    $r \leftarrow Y^q - \big(\sum_{j \in \mathcal{J}} a_j y_{\eta_j, \varpi_j}(\cdot - \mu_j)\big)^q$
    $\theta \leftarrow \|r\|_2$
    if $\|r\|_2 \geq \lambda\theta$ **then**
      restore values from previous iteration
      break
  for $\eta = 0, \dots, N_{\mathrm{pat}} - 1$ **do**
    $A[\eta] \leftarrow A[\eta] + \sum_{\eta_j = \eta} a_j$
  return $\mathcal{J}, a_{\mathcal{J}}, \mu_{\mathcal{J}}, \varpi_{\mathcal{J}}, \eta_{\mathcal{J}}, A$

Figure 6. Sparse identification algorithm

The sel_xcorr function in Figure 7 is used in the separation. It selects up to N_{pre} patterns based on cross-correlation, and it computes their discrete amplitudes and shifts. In the implementation, this is accelerated via the FFT convolution theorem. The sel_peaks function in Figure 8 ignores the patterns, and it simply returns the N_{pre} largest local maxima with dominance N_{dom} (typically, $N_{\mathrm{dom}} = 3$).

The dictionary learning algorithm (Figure 9) is largely identical to the original formulation of Adam (with values $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, and a step-size of $\kappa = 10^{-3}$), except that v_2 is averaged over all the harmonics for one instrument. It counts the number of training iterations $\tau[\eta]$ for each instrument η individually. The dictionary is initialized by the function in Figure 10, which creates a new dictionary column with random values. The function in Figure 11 removes seldom-used instruments in the dictionary by comparing their average amplitude, but with a head start which is half the length of the pruning interval: $\tau_0 = N_{\mathrm{prn}}/2$.

The logspect function in Figure 12 takes an STFT magnitude spectrogram and applies the sparse pursuit algorithm in order to convert it into a log-frequency spectrogram with a height of $m = 1024$. Finally, the separate function in Figure 13 performs the overall separation procedure. It renews the dictionary every $N_{\mathrm{prn}} = 500$ steps.

function sel_xcorr($r, y_0, \dots, y_{N_{\mathrm{pat}}-1}, \mathcal{J}, N_{\mathrm{pre}}$)
  $\rho[\mu, \eta] \leftarrow \sum_{i=0}^{m-1} r[i] \left(y_{\eta, \varpi_{\mathrm{nil}}}(i - \mu)\right)^q / \|y_{\eta, \varpi_{\mathrm{nil}}}[\cdot]^q\|_2$
    for $\mu = 0, \dots, m-1$ and $\eta = 0, \dots, N_{\mathrm{pat}} - 1$
  $(\mu_{\mathcal{J}}, \eta_{\mathcal{J}}) \leftarrow \arg\mathrm{sort}_{(\mu,\eta)}(-\rho[\mu, \eta])[: N_{\mathrm{pre}}]$
  $a_{\mathcal{J}} \leftarrow (\rho[\mu_{\mathcal{J}}, \eta_{\mathcal{J}}] / \|y_{\eta_{\mathcal{J}}, \varpi_{\mathrm{nil}}}[\cdot]^q\|_2)^{1/q}$
  $\mathcal{J} \leftarrow \{j \in \mathcal{J} : a_j > 0\}$
  return $a_{\mathcal{J}}, \mu_{\mathcal{J}}, \eta_{\mathcal{J}}$

Figure 7. Selector function based on cross-correlation

function sel_peaks($r, \_, \mathcal{J}, N_{\mathrm{pre}}$)
  $\mu_{\mathcal{J}} \leftarrow \arg\mathrm{sort}_{\mu}$
    $\{a_\mu : r[\mu] \geq r[\mu + k],\ |k| \leq N_{\mathrm{dom}}\}[: N_{\mathrm{pre}}]$
  $\mathcal{J} \leftarrow \{j \in \mathcal{J} : r[j] > 0\}$
  return $r_{\mathcal{J}}, \mu_{\mathcal{J}}, 0$

Figure 8. Selector function based on peaks

function adam(D, τ, v_1, v_2, g)
  for $\eta = 0, \dots, N_{\mathrm{pat}} - 1$ **do**
    $\tau[\eta] \leftarrow \tau[\eta] + 1$
    $v_1[\cdot, \eta] \leftarrow \beta_1 \cdot v_1[\cdot, \eta] + (1 - \beta_1) \cdot g[\cdot, \eta]$
    $v_2[\eta] \leftarrow \beta_2 \cdot v_2[\eta] + (1 - \beta_2) \cdot \mathrm{mean}(g[\cdot, \eta]^2)$
    $\hat{v}_1[\cdot, \eta] \leftarrow v_1[\cdot, \eta] / (1 - \beta_1^{\tau[\eta]})$
    $\hat{v}_2[\eta] \leftarrow v_2[\eta] / (1 - \beta_2^{\tau[\eta]})$
    $D[\cdot, \eta] \leftarrow D[\cdot, \eta] - \kappa \cdot \hat{v}_1[\cdot, \eta] / (\sqrt{\hat{v}_2[\eta] + \varepsilon})$
    $D[\cdot, \eta] \leftarrow \max\big(0, \min\big(1, D[\cdot, \eta]\big)\big)$
  return D, τ, v_1, v_2

Figure 9. Dictionary learning function

function init()
  $e \leftarrow \mathrm{Par}(1, 2)$
  for $h = 1, \dots, N_{\mathrm{har}}$ **do**
    $d[h] \leftarrow \mathcal{U}[0, 1)$
    $d[h] \leftarrow d[h] / h^e$
  return $d[\cdot]$

Figure 10. Dictionary initialization function

function renew(A, D, τ)
  $\mathcal{I} \leftarrow \big(\arg\mathrm{sort}_{\eta \in \{0,\dots,N_{\mathrm{pat}}-1\}} A[\eta] / (\tau[\eta] - \tau_0)\big)$
    $[0, \dots, N_{\mathrm{ins}}]$
  $\tau[\mathcal{I}^{\complement}] = 0,\ v_1[\cdot, \mathcal{I}^{\complement}] = 0,\ v_2[\mathcal{I}^{\complement}] = 0,\ A[\mathcal{I}^{\complement}] = 0$
  for $\eta \in \mathcal{I}^{\complement}$ **do**
    $D[\cdot, \eta] \leftarrow$ init()
  return $\mathcal{I}, \tau, v_1, v_2, A, D$

Figure 11. Dictionary renewal function

function logspect($Z, N_{\mathrm{pre}}, N_{\mathrm{spr}}$)
  for $t = 0, \dots, n-1$ **do**
    $\mathcal{J}_t, a_{\mathcal{J}_t,t}, \mu_{\mathcal{J}_t,t}, \varpi_{\mathcal{J}_t,t}, \eta_{\mathcal{J}_t,t}, \_$
      $\leftarrow$ pursuit($[1], Z[\cdot,t]$, sel_peaks, $1, N_{\mathrm{spr}}, \_$)
  $U[\alpha,t] \leftarrow \sum_{j,h} a_{j,h,t} \exp(-(\alpha-\alpha(\mu_{\mathcal{J}_t,t}))^2/(2F^2\sigma_{j,t}^2))$
    for $\alpha = 0, \dots, m-1$, $t = 0, \dots, n-1$
  return U

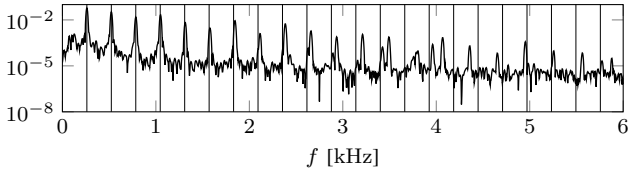
Figure 12. Log-spectrogram generation function

function separate($U, N_{\mathrm{pre}}, N_{\mathrm{spr}}$)
  for $\eta = 0, \dots, N_{\mathrm{pat}} - 1$ **do**
    $D[\cdot,\eta] \leftarrow$ init()
    $\tau[\eta] \leftarrow 0$, $v_1[\cdot,\eta] \leftarrow 0$, $v_2[\eta] \leftarrow 0$, $A[\eta] \leftarrow 0$
  loop a multiple of N_{prn} times
    $t \leftarrow \mathrm{random}(\{0, \dots, n-1\})$
    $\mathcal{J}, a_{\mathcal{J}}, \mu_{\mathcal{J}}, \varpi_{\mathcal{J}}, \eta_{\mathcal{J}}, A$
      $\leftarrow$ pursuit($D, U[\cdot,t]$, sel_xcorr, $1, N_{\mathrm{spr}}, A$)
    $g \leftarrow \nabla_D R(D, Y, a_{\mathcal{J}}, \mu_{\mathcal{J}}, \varpi_{\mathcal{J}}, \eta_{\mathcal{J}})$
    $D, \tau, v_1, v_2 \leftarrow$ adam(D, τ, v_1, v_2, g)
    if $\min(\tau) \bmod N_{\mathrm{prn}} = 0$ **then**
      $\mathcal{I}, \tau, v_1, v_2, A, D \leftarrow$ renew(A, D, τ)
  for $t = 0, \dots, n-1$ **do**
    $\mathcal{J}_t, a_{\mathcal{J}_t,t}, \mu_{\mathcal{J}_t,t}, \varpi_{\mathcal{J}_t,t}, \eta_{\mathcal{J}_t,t}, \_$
      $\leftarrow$ pursuit($D[\cdot,\mathcal{I}], U[\cdot,t]$, sel_xcorr, $1, N_{\mathrm{spr}}, \_$)
  return $\{\mathcal{J}_t, a_{\mathcal{J}_t,t}, \mu_{\mathcal{J}_t,t}, \varpi_{\mathcal{J}_t,t}, \eta_{\mathcal{J}_t,t}$
    $: t = 0, \dots, n-1\}$

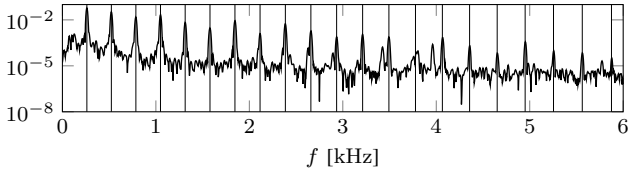
Figure 13. Dictionary learning and separation function

Inharmonicity of a Piano Tone

We show the necessity to include the inharmonicity parameter in the model of the harmonics in Figure 14. When setting $b = 0$, the frequency of higher harmonics differs strongly from the prediction, while with the correct inharmonicity, they match the model very accurately.



(a) with $b = 0$

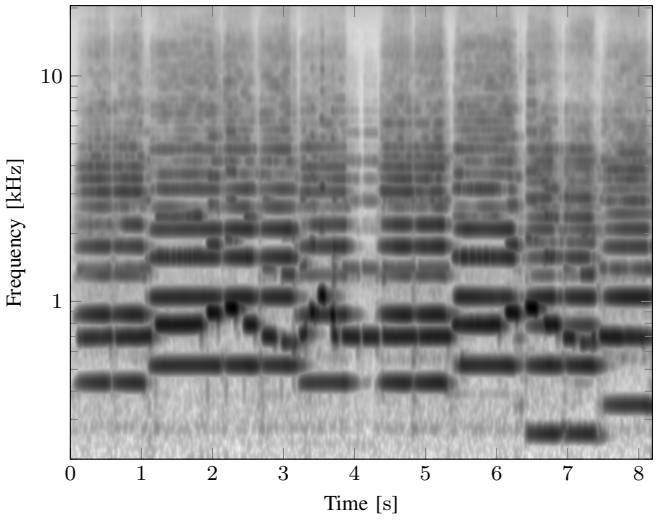


(b) with $b = 3.25 \times 10^{-4}$

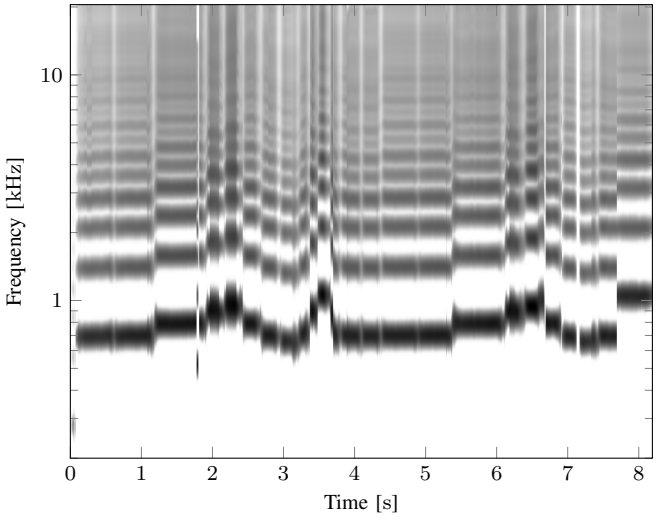
Figure 14. Modeling the harmonics of a piano tone. The sample was taken from a Yamaha P-120 electronic piano, playing the c' tone.

Benefits Over the Mel Spectrogram

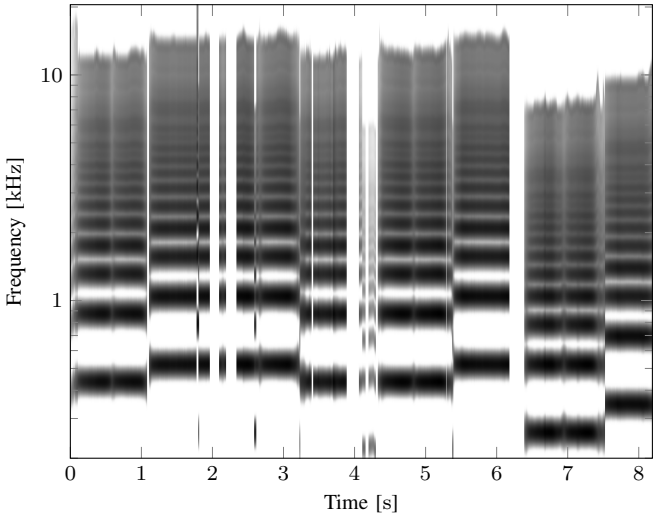
In Figure 3, we compared our log-spectrogram that was computed via the sparse pursuit method to the mel spectrogram



(a) Mel spectrogram for the original sample



(b) Synthesized recorder track



(c) Synthesized violin track

Figure 15. Mel spectrogram for the recorded piece and log-frequency spectrograms for the synthesized tracks that were generated based on the mel spectrogram.

Table IV
PERFORMANCE MEASURES FOR THE SEPARATION OF RECORDER AND VIOLIN ON THE MEL SPECTROGRAM, WITH SPECTRAL MASKING.

Mask	Instrument	Measure	Best	Mean	Stdev
No	Recorder	SDR	10.6	7.7	5.2
		SIR	31.6	28.4	3.7
		SAR	10.7	7.7	5.2
	Violin	SDR	5.6	2.9	4.0
		SIR	23.0	18.2	8.1
		SAR	5.7	3.6	3.0
Yes	Recorder	SDR	13.9	9.9	6.1
		SIR	31.2	27.9	3.8
		SAR	14.0	10.0	6.1
	Violin	SDR	9.2	6.0	4.5
		SIR	21.8	16.3	8.2
		SAR	9.4	7.5	2.3

and the constant-Q transform. We concluded in Section III that as the CQT uses windows of different length for different frequencies, it is not a good choice for our dictionary representation.

The mel spectrogram does not have this particular problem, but the Heisenberg uncertainty principle constrains the time-log-frequency resolution according to the lowest frequency to be represented. In Figure 3a, we cut the spectrogram at $530\,\mathrm{Hz}$ (which corresponds to $577\,\mathrm{Hz}$ when compensating for the different sampling frequency), but for our sample with recorder and violin, this is not sufficient, as it contains notes as low as c'. Thus, we chose the lowest frequency as $200\,\mathrm{Hz}$, sacrificing some resolution.

We computed the mel spectrogram on this sample and ran the separation algorithm 10 times with $N_{\mathrm{itr}} = 100000$ training iterations in order to obtain a fair comparison. The performance figures are given in Table IV, and the results from the best-case run with a random seed of 2 are displayed in Figure 15. It can be seen that the performance does not reach what we achieved with a spectrogram generated via the sparse pursuit method (cf. Figure 4 and Table I).

We thus conclude that our use of the sparse pursuit algorithm for generating a log-frequency spectrogram provides a notable benefit for the subsequent processing.

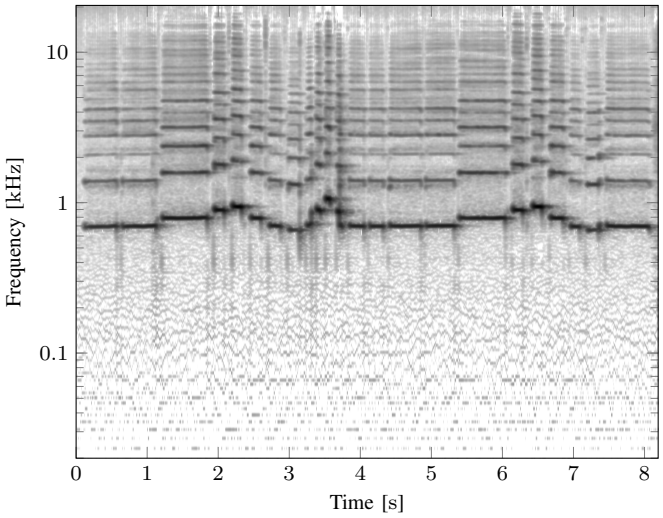
LOG-FREQUENCY SPECTROGRAMS OF THE INSTRUMENT TRACKS

For additional comparison, we computed the log-frequency spectrograms of the original (Figure 16) and the computed (Figure 17) instrument tracks.

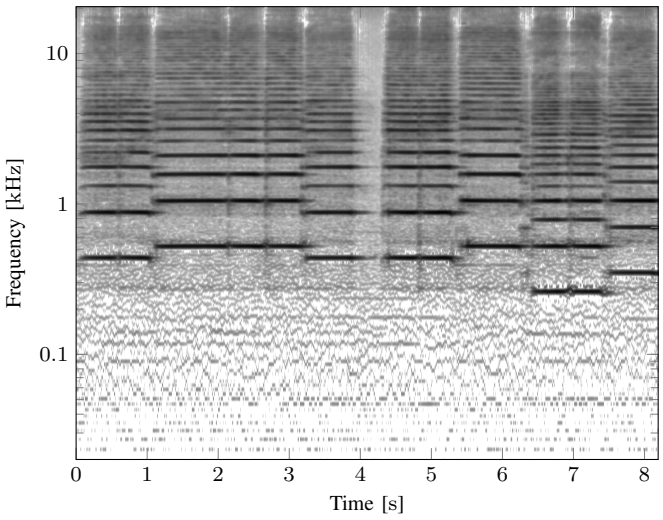
It should be noted, however, that these spectrograms are not used anywhere in the computation or evaluation process, and due to artifacts from the sparse pursuit algorithm, they are not an accurate representation of the time-domain signal.

Nevertheless, two effects can be seen when comparing Figure 17 to Figure 4:

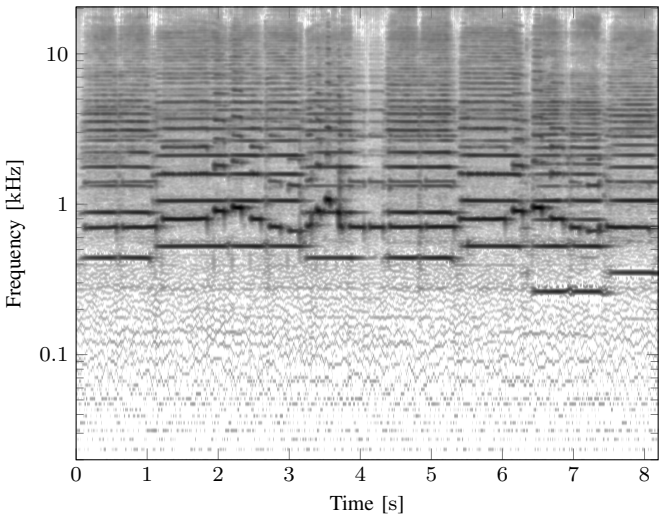
1) Due to spectral masking, the harmonics now have different intensities.
2) The Griffin-Lim phase reconstruction algorithm *smoothes* some of the artifacts that were introduced by the sparse pursuit algorithm. This is because not every two-dimensional image is actually a valid spectrogram that corresponds to an audio signal; instead, the Griffin-Lim algorithm aims to find an audio signal whose spectrogram is as close as possible to the given image, and it uses the phase of the original mixed sample as the initial value.



(a) Recorder

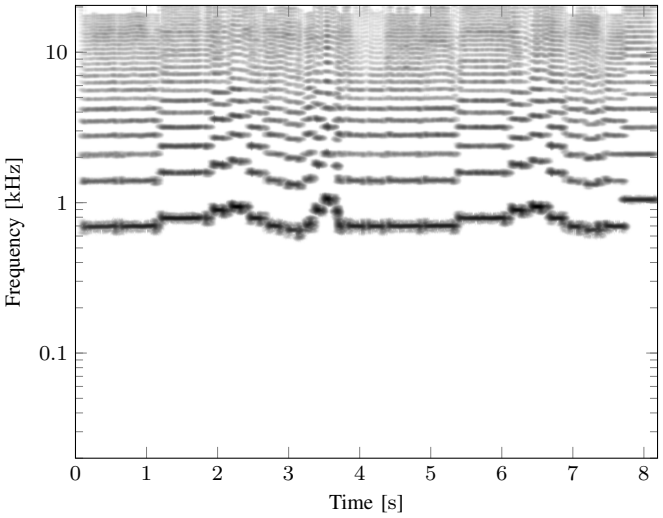


(b) Violin

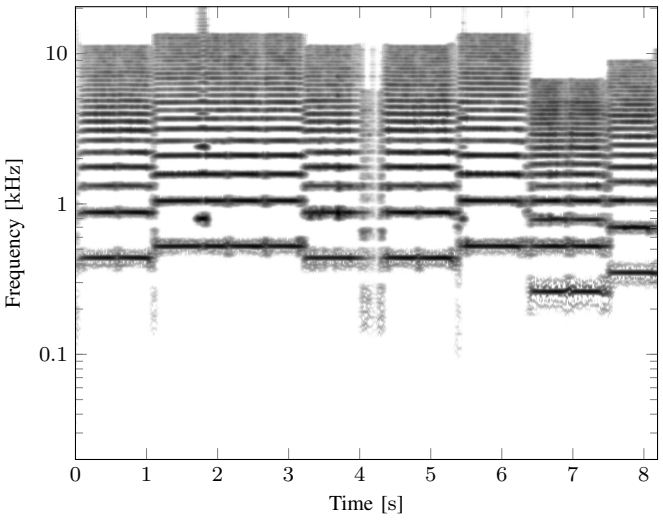


(c) Mixture (copy of Figure 4a)

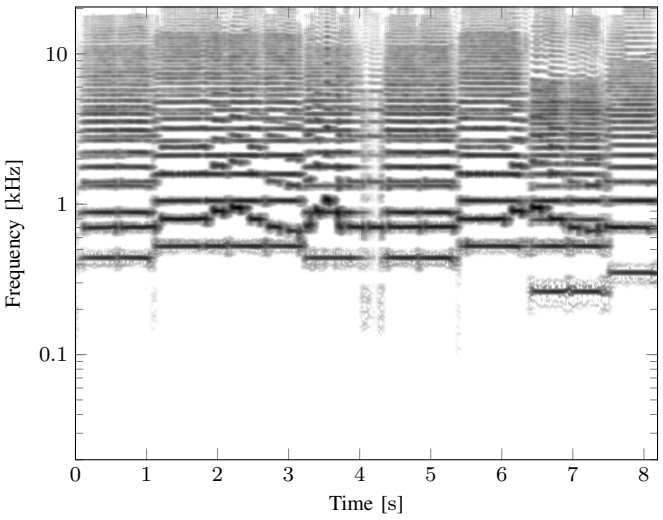
Figure 16. Sparsity-derived log-frequency spectrograms of the original instrument samples



(a) Recorder



(b) Violin



(c) Mixture

Figure 17. Sparsity-derived log-frequency spectrograms of the separated audio tracks and its mixture