Auditory-Inspired Sparse Representation of Audio Signals

Ramin Pichevar, Hossein Najaf-Zadeh, Louis Thibault, and Hassan Lahdili
Communications Research Centre, 3701 Carling Ave., Ottawa, Canada

Abstract

This article deals with the generation of auditory-inspired spectro-temporal features aimed at audio coding. To do so, we first generate sparse audio representations we call spikegrams, using projections on gammatone/gammachirp kernels that generate neural spikes. Unlike Fourier-based representations, these representations are powerful at identifying auditory events, such as onsets, offsets, transients and harmonic structures. We show that the introduction of adaptiveness in the selection of gammachirp kernels enhances the compression rate compared to the case where the kernels are non-adaptive. We also integrate a masking model that helps reduce bitrate without loss of perceptible audio quality. We finally propose a method to extract frequent audio objects (patterns) in the aforementioned sparse representations. The extracted frequency-domain patterns (audio objects) help us address spikes (audio events) collectively rather than individually. When audio compression is needed, the different patterns are stored in a small codebook that can be used to efficiently encode audio materials in a lossless way. The approach is applied to different audio signals and results are discussed and compared. This work is a first step towards the design of a high-quality auditory-inspired “object-based” audio coder.

1. Introduction

Non-stationary and time-relative structures such as transients, timing relations among acoustic events, and harmonic periodicities provide important cues for different types of audio processing techniques including audio coding, speech recognition, audio localization and auditory scene analysis. Obtaining these cues is a difficult task. The most important reason why it is so difficult is that most approaches to signal representation/analysis are block-based, i.e. the signal is processed piecewise in a series of discrete blocks. Therefore, transients and non-stationary periodicities in the signal can be temporally smeared across blocks [Smith and Lewicki, 2005]. Moreover, large changes in the representation of an acoustic event can occur depending on the arbitrary alignment of the processing blocks with events in the signal. Signal analysis techniques such as windowing or the choice of the transform can reduce these effects, but it
would be preferable if the representation was insensitive to signal shifts. Shift-
invariance alone, however, is not a sufficient constraint on designing a general
sound processing algorithm. A desirable representation should capture the un-
derlying 2D-time-frequency structures, so that they are more directly observable
and well represented at low bit rates [Smith and Lewicki, 2005]. These structures
must be easily extractable as audio objects for further processing in coding,
recognition, etc.

The aim of this article is to propose an auditory-inspired coding scheme,
which includes many characteristics of the auditory pathway such as sparse
coding, masking, audio object extraction, and recognition (see Fig. 1). More
specifically, we introduce an adaptive approach to the extraction of sparse codes
and show by objective and subjective tests that the adaptive approach out-
performs the classical matching pursuit approach (as described in [Smith and
Lewicki, 2005]). We also show that the addition of a masking model to the
classical MP can enhance the perceived quality of the reconstructed signal. We
call this new technique Perceptual Matching Pursuit (PMP). Finally, we pro-
pose an audio object extraction algorithm based on temporal episode discovery
[Patnaik et al., 2008]. The audio object extraction helps us address sparse codes
collectively rather than individually and consequently save in coding bitrate.

In the next section we will give a brief survey of different coding schemes to
justify our choices for our proposed approach.

2. Coding Schemes

In this section we compare different coding approaches and will justify our
choice of using sparse overcomplete codes. For this purpose, we briefly com-
pare three approaches: block-based coding, filterbank coding, and overcomplete
representations.

2.1. Block-Based Coding

Most of the signal representations used in speech and audio coding are block
based (i.e., DCT, MDCT, FFT). In the block-based coding scheme, the signal
is processed piecewise in a series of discrete blocks, causing temporally smeared
transients and non-stationary periodicities. Moreover, large changes in the rep-
resentation of an acoustic event can occur depending on the arbitrary alignment
of the processing blocks with events in the signal. Signal analysis techniques
such as windowing or the choice of the transform can reduce these effects, but
it would be preferable if the representation was insensitive to signal shifts.

2.2. Filterbank-Based Shift-Invariant Coding

In the filterbank design paradigm, the signal is continuously applied to the
filters of the filterbank and its convolution with the impulse responses are com-
puted. Therefore, the outputs of these filters are shift invariant. This repre-
sentation does not have the drawbacks of block-based coding mentioned above,
such as time variance. However, filterbank analysis is not sufficient for designing a general sound processing algorithm. Another important aspect not taken into account in this paradigm is coding efficiency or, equivalently the ability of the representation to capture underlying structures in the signal. A desirable code/representation should reduce the information redundancy from the raw signal so that the underlying structures are more directly observable. However, convolutonal representations (i.e., filterbank design) increase the dimensionality of the input signal.

2.3. Overcomplete Shift-Invariant Representations

In an overcomplete representation, the number of basis vectors (kernels) is greater than the real dimensionality (number of non-zero eigenvalues in the covariance matrix of the signal) of the input. The approach consists of matching the best kernels to different acoustic cues using different convergence criteria such as the residual energy. However, the minimization of the energy of the residual (error) signal is not sufficient to get an overcomplete representation of an input signal. Other constraints such as sparseness must be considered in order to have a unique solution [Graham and Field, 2006]. Overcomplete representations have been advocated because they have greater robustness in the presence of noise [Graham and Field, 2006]. They are also a way to maximize information transfer, when different regions/objects of the underlying signal have strong correlations [Graham and Field, 2006]. In other terms, the peakiness of values can be exploited efficiently in entropy coding. In order to find the “best matching kernels”, matching pursuit is used.

2.4. Generating Overcomplete Representations with Matching Pursuit (MP)

In mathematical notations, the signal $x(t)$ can be decomposed into the overcomplete kernels as follow

$$
x(t) = \sum_{m=1}^{M} \sum_{i=1}^{n_m} a_i^m g_m(t - \tau_i^m) + r_x(t) \tag{1}
$$

where $\tau_i^m$ and $a_i^m$ are the temporal position and amplitude of the $i$-th instance of the kernel $g_m$, respectively. The notation $n_m$ indicates the number of instances of $g_m$, which need not be the same across kernels. In addition, the kernels are not restricted in form or length.

In order to find adequate $\tau_i^m$, $a_i^m$, and $g_m$ matching pursuit can be used [Mallat and Zhang, 1993]. In this technique the signal $x(t)$ is decomposed over a set of kernels so as to capture the structure of the signal. The approach consists of iteratively approximating the input signal with successive orthogonal projections onto some basis. The signal can be decomposed into

$$
x(t) = \langle x(t), g_m \rangle + r_x(t) \tag{2}
$$

where $\langle x(t), g_m \rangle$ is the inner product between the signal and the kernel and is equivalent to $a_i^m$ in Eq. 1. $r_x(t)$ is the residual signal. Note that kernels $g_m$ have unit norm.
It can be shown [Goodwin and Vetterli, 1999] that the computational load of the matching pursuit can be reduced, if one saves values of all correlations in memory or finds an analytical formulation for the correlation given specific kernels.

3. A New Paradigm for Audio Coding

3.1. The Bio-Inspired Audio Coder

The analysis/synthesis part of our universal audio codec is based on the generation of auditory-inspired sparse 2-D representations of audio signals, dubbed as spikegrams. The spikegrams are generated by projecting the signal onto a set of overcomplete adaptive gammachirp (gammatones with additional tuning parameters) kernels (see section 3.2.2). The adaptiveness is a key feature we introduced in Matching Pursuit (MP) to increase the efficiency of the proposed method (see section 3.2.2) and is represented by the block “Greedy Adaptive Selection of Additional Parameters” in Fig. 1. A masking model (Time-frequency plane masking in Fig. 1) is applied to the spikegrams to remove inaudible spikes (see section 4). In addition a differential encoder of spike parameters based on graph theory is proposed in [Pichevar et al., 2008a]. The quantization of the spikes is proposed in [Pichevar et al., 2008b]. The block diagram of all the building blocks of the receiver and transmitter of our proposed universal audio coder is depicted in Fig. 1. Our focus in this article will be on the adaptive MP, Time-frequency masking, and the frequent pattern discovery blocks.

![Figure 1: Block diagram of the Universal Bio-Inspired Audio Coder.](image)

3.2. Generation of the spike-based representation

We propose an auditory sparse and overcomplete representation suitable for audio processing. In this paradigm, the signal is decomposed into its constituent parts (kernels) by a matching pursuit algorithm. We use gammatone/gammachirp filterbanks as projection basis as proposed in [Smith and
Lewicki, 2005] [Smith and Lewicki, 2006]. Note that our approach is different from other works in the literature (i.e., [Feldbauer et al., 2005]) in which gammatones are used as a filterbank and not as kernels for the generation of sparse representations based on matching pursuit. It is also different from approaches where gammatones are used in the context of parameter coding (i.e., [Christensen and van de Par, 2006], where gammatones are used as envelopes to sinusoids in the context of transient coding. The advantage of using asymmetric kernels such as gammatone/gammachirp atoms is that they do not create pre-echos at onsets [Goodwin and Vetterli, 1999]. However, very asymmetric kernels such as damped sinusoids [Goodwin and Vetterli, 1999] are not able to model harmonic signals suitably. On the other hand, gammatone/gammachirp kernels have additional parameters that control their attack and decay parts (degree of symmetry), which are modified suitably according to the nature of the signal in our proposed technique. As described above, the approach is an iterative one. We will compare two variants of the technique. The first variant, which is non-adaptive, is roughly similar to the general approach used in [Smith and Lewicki, 2006], which we applied to the specific task of audio coding. However, the second adaptive variant is a novel one, which takes advantage of the additional parameters of the gammachirp kernels [Irino and Patterson, 2001],[Irino and Patterson, 2006]. Some details on each variant are given below.

3.2.1. Non-Adaptive Paradigm

In the non-adaptive paradigm, only gammatone filters are used. The impulse response of a gammatone filter is given by

\[ g(f_c, t) = t^{(l-1)}e^{-2\pi bt}\cos(2\pi f_c t) \quad t > 0, \]

where \( f_c \) is the center frequency of the filter, distributed on Equivalent Rectangular Bandwidth (ERB) scales between 65 Hz and 14 kHz as described in [Patterson and Moore, 1986], and \( l = 4 \) for the gammatone. At each step (iteration), the signal is projected onto the gammatone kernels (with different center frequencies and different time delays). The center frequency and time delay that give the maximum projection are chosen and a spike with the value of the projection is added to the “auditory representation” at the corresponding center frequency and time delay (see Fig. 2). The signal is decomposed into the projections on gammatone kernels plus a residual signal \( r_x(t) \) (see Eqs. 1 and 2).

3.2.2. Adaptive Paradigm

In [Strahl and Mertins, 2008], it has been shown that the optimal gammatone parameters, found by fitting to the human auditory system, do not match the parameters estimated from English speech signals. From those observations, one can conclude that the optimal gammatone parameters differ for different type of signals. Therefore, we decided to adaptively modify gammatone parameters as described below. Furthermore, we introduced an additional parameter (the chirp factor) that can be adjusted at each MP iteration. Gammatones with this
additional parameter are called gammachirps [Irino and Patterson, 2001]. In
the adaptive paradigm, gammachirp kernels are used. The impulse response of
a gammachirp filter with the corresponding tuning parameters \((b, l, c)\) and start
time \(t_0\) is given below:

\[
g(f_c, t, b, l, c; t_0) = (t - t_0)^{l-1}e^{-2\pi b(t-t_0)}\cos[2\pi f_c(t - t_0) + \ln(t - t_0)]u(t - t_0),
\]
where \(u(\cdot)\) is the Heaviside step function. It has been shown that the ga-
machirp filters minimize the scale/time uncertainty [Irino and Patterson, 2001].
In our approach the chirp factor \(c\), \(l\), and \(b\) are found adaptively at each step.
The chirp factor \(c\) allows us to slightly modify the instantaneous frequency of the
kernels, \(l\) and \(b\) control the attack and decay of the kernels. However, searching
the three parameters in the parameter space is a very computationally intensive
task. Therefore, we use a suboptimal search [Gribonval, 2001] in which, we use
the same gammatone filters as the ones used in the non-adaptive paradigm with
values of \(l\) and \(b\) given in [Irino and Patterson, 2001]. We then find the center
frequency and start time \((t_0)\) of the best gammatone matching filter. We also
keep the second best frequency (gammatone kernel) and start time.

\[
G_{max_1} = \arg\max_{f, t_0} \{|< r, g(f, t, b, l, c; t_0) >|\}, \quad g \in G
\]
\[
G_{max_2} = \arg\max_{f, t_0} \{|< r, g(f, t, b, l, c; t_0) >|\}, \quad g \in G - G_{max_1}
\]

For the sake of simplicity, we use \(f\) instead of \(f_c\) in Eqs. 5 to 9. Experimental
results (see Fig. 5) show that the chirp factor has a greater impact on conver-
gence than the attack and decay factors \((l\) and \(b)\). Therefore, the chirp factor
is optimized before the other two factors. In other words, we keep only the set
of the best two kernels in step one, and try to find the best chirp factor given
\(g \in G_{max_1} \cup G_{max_2}\).

\[
G_{max_c} = \arg\max_{c} \{|< r, g(f, t, b, l, c; t_0) >|\}.
\]
We then use the information found in the second step to find the best \( b \) for \( g \in G_{max} \) in Eq. 8, and finally find the best \( l \) among \( g \in G_{max} \) in Eq. 9.

\[
G_{maxb} = \arg\max_b \{ |\langle r, g(f, t, b, l, c; t_0) \rangle| \} \quad (8)
\]

\[
G_{maxl} = \arg\max_l \{ |\langle r, g(f, t, b, l, c; t_0) \rangle| \} \quad (9)
\]

In this work, the chirp values range from -4 to 2 with a quantization step of 0.2. Furthermore, the search space for \( l \) is between 3.5 and 4 (quanta of 0.02). Finally, \( b \) varies between 0.7ERB\((f_c)\) and 1.3ERB\((f_c)\) with 0.01ERB\((f_c)\) steps. Therefore, six parameters are extracted in the adaptive technique for the “auditory representation”: optimal center frequencies, chirp factors \( (c_{opt}) \), time delays, spike amplitudes, optimal \( b \) \( (b_{opt}) \), and optimal \( l \) \( (l_{opt}) \). The last two parameters control the attack and the decay slopes of the kernels. Although, there are additional parameters in this second variant, as shown later, the adaptive technique contributes to better coding gains. The reason for this is that we need a much smaller number of atoms/kernels and a smaller number of iterations to achieve the same audio quality. We compare in detail the two approaches in section 3.4.

3.3. Speeding-up the Optimization Process

In the previous section, we performed a pseudo-optimal search of the parameter space to find the optimal chirp factor, attack, and decays. Here, we propose a speedup to the optimization of the chirp factor. Upon finding the center frequency and start time \( (t_0) \) of the best gammatone matching filter (Eqs. 5 and 6), we compute the local correlation of the gammachirp kernel with the residual. In other words, we compute:

\[
\Gamma(c) = \langle r, g(f, n, b, l, c; n_0) \rangle \quad (10)
\]

\[
= \sum_{n=n_0+1}^{N-1+n_0} r(nT_s)T_s(n-n_0)\{2\pi fT_s(n-n_0) + c\ln[T_s(n-n_0)]\}^2 e^{-2\pi bT_s(n-n_0)} \cos\{2\pi fT_s(n-n_0) + c\ln[T_s(n-n_0)]\},
\]

where \( n \) is the discretized time and \( n_0 \) is the discretized \( t_0 \), and \( N \) is the length of the gammachirp. \( T_s \) is the sampling period. Fig. 3 shows that \( \Gamma(c) \), the dot product function, is an oscillatory curve. As such, we do the optimization procedure in two steps to avoid a local maximum (minimum). First the location of the maximum of the correlation envelope is found. Then, the found value is used as the initial value for the Newton-Raphson method to determine a globally optimal chirp factor. A similar approach can be applied to find the optimal \( l \) and the optimal \( b \).

3.4. Comparison of Adaptive and Non-Adaptive Paradigms

In this section we compare the performance of the adaptive and non-adaptive schemes. The non-adaptive method is an implementation of [Smith and Lewicki, 2005][Smith and Lewicki, 2006] and is used as a baseline in our experiments.
We show in the remainder of this section that our adaptive scheme outperforms the standard method proposed by Smith and Lewicki [Smith and Lewicki, 2005][Smith and Lewicki, 2006]. Results and a comparison of the two different schemes in terms of bitrate and number of spikes extracted for high quality (scale 4 on ITU-R impairment scale [Bech and Zakharov, 2006]) are given in Table 1. Results are obtained using the “Matching Pursuit”, “Greedy Adaptive Selection of Additional Parameters”, “Spikegram Generation”, and “Arithmetic Coding” blocks in Fig. 1. We used a 128-level uniform quantization and arithmetic coding for spike amplitudes to generate values in Table 1 (for a more efficient optimization algorithm see [Pichevar et al., 2008b]). With the adaptive scheme, we observe an average drop of 45% in the bitrate compared to the non-adaptive approach. We also computed the spike gain, which is the average number of spikes required for a given quality for \( N \) time-domain samples. In other words, it shows the ratio between the number of spikes and the number of time-domain samples. The spike gain decreases drastically when the adaptive paradigm is used as well. Fig. 4 compares the adaptive to the non-adaptive approach for different numbers of cochlear channels (the number of center frequencies used in the gammatone kernels). Furthermore, Figs. 6, 7, and 8 compares the evolution of the quality of a 10-second speech signal for the adaptive and non-adaptive cases. In Fig. 7, the objective PEAQ scores [Thiede et al., 2000] (obtained by CRC-SEAQ 1) are plotted, while for Fig. 8 the mean scores of 6 trained listeners are plotted.

Table 2 compares PEAQ results for different types of signals at the same average number of spikes per sample (0.22 spikes/sample). The objective difference score for two of the signals (piano and violin) are more than 2 units between the adaptive and non-adaptive cases.

Table 1: Comparison of the adaptive and non-adaptive schemes for spike generation for three different audio signals. The average saving in bitrate over all materials is around 45%. N is the signal length (number of samples in the signal). Bitrates include the cost to transmit the additional $b$, $l$, and $c$ parameters.

<table>
<thead>
<tr>
<th></th>
<th>Speech</th>
<th>Castanet</th>
<th>Percussion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adapt.</td>
<td>Non-Adapt.</td>
<td>Adapt.</td>
</tr>
<tr>
<td>Number of spikes</td>
<td>10492</td>
<td>35208</td>
<td>6510</td>
</tr>
<tr>
<td>Spike gain</td>
<td>0.13N</td>
<td>0.44N</td>
<td>0.08N</td>
</tr>
<tr>
<td>Bitrate (bit/sample)</td>
<td>1.98</td>
<td>3.07</td>
<td>1.54</td>
</tr>
</tbody>
</table>

4. Masking

In this section, we introduce a Perceptual Matching Pursuit (PMP) algorithm for audio processing. A masking model has been developed and integrated into the matching pursuit algorithm to account for the characteristics of the hearing system. By doing so, only an audible kernel is extracted at each iteration. Moreover, contrary to the matching pursuit algorithm, PMP will stop decomposing an audio signal once there is no audible part left in the residual.

One approach to incorporate auditory masking into a matching-pursuit-based audio encoder is to extract many kernels followed by passing the extracted kernels through a masking module to eliminate the inaudible kernels [Pichevar et al., 2007]. In that approach several kernels (including inaudible ones) that cause unnecessary computational load are extracted. Another approach is to give perception-based weights to different kernels based on their perceptual im-
Figure 5: Analysis of the effect of the adaptation of each of the three parameters (the chirp factor \( c \), \( b \), and \( l \)) on a violin signal. For each curve, one parameter is adapted and the other two are kept constant. Curves for the case where all three parameters are adapted and the case where no adaptation occurs are plotted for the sake of comparison.

Figure 6: Comparison of SNR vs. number of iterations for a 10-second speech signal in the adaptive and non-adaptive cases.

Importance [Verma and Meng, 1999] [Heusdens et al., 2001]. This method will result in better audio quality. However, it does not suggest any stop criterion once there is no audible part left in the audio signal. In addition, works in [Verma and Meng, 1999] [Heusdens et al., 2001] do not implement any temporal masking (they deal with simultaneous masking). Our approach is to progressively create a Time-Frequency (TF) masking pattern to determine a masking threshold at all time indexes and frequencies. Once no kernel having a magnitude above the masking threshold can be extracted, the decomposition stops and the audio signal can be reconstructed from the extracted kernels without loss in quality. The accuracy of the masking model plays a key role in the quality of the reconstructed audio signal.
4.1. Time-Frequency Masking Model

Due to similarities between gammatones and gammachirps and for the sake of convenience, gammatone functions are employed to develop the masking model. We set the initial level for the global masking pattern in critical band $k$ (at all time instances) $M_{TOT}(n, k)$ to the absolute threshold of hearing and progressively add to it the masking effects caused by extracted kernels. The following situations for the masking pattern caused by an extracted kernel are considered.

4.1.1. Forward Masking in the Same Critical Band

This is the situation when a maskee starts with a small delay after the masker. We compute the masking effect of a spike (masker) on other spikes (maskees) generated in a time window following the masker spike. For forward masking, we assume a linear relation between the masking threshold (in dB) and the logarithm of the time delay between the masker and the maskee in
Table 2: Objective difference grade scores based on PEAQ for different types of 10-second long signals sampled at 44.1kHz. A score between 0 and -1 corresponds to transparent quality.

<table>
<thead>
<tr>
<th>Signal Type</th>
<th>PEAQ</th>
<th>Non-Adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Violin</td>
<td>-0.7</td>
<td>-2.9</td>
</tr>
<tr>
<td>Castanet</td>
<td>-0.3</td>
<td>-0.8</td>
</tr>
<tr>
<td>Piano</td>
<td>-0.1</td>
<td>-2.1</td>
</tr>
<tr>
<td>Speech</td>
<td>-0.5</td>
<td>-1.5</td>
</tr>
</tbody>
</table>

milliseconds [Zwicker, 1984]. Since the effective duration of forward masking depends on the masker duration [Zwicker and Fastl, 1990], we empirically define an effective duration $\rho_k$ (in ms) for forward masking in each critical band as follows:

$$\rho_k = 100 \arctan(d_k)$$  \hspace{1cm} (11)

where $d_k$ (in ms) denotes the duration of the kernel in band $k$, defined as the time interval between the points on the temporal envelope of the kernel where the amplitude drops by 90%. Note that according to psychoacoustical data, effective duration of forward masking is less than 200 ms should the auditory stimulus be long enough. Eq. 11 takes into account the limited length of the kernels and accordingly shortens the effective duration of auditory forward masking effects.

If a spike has a level below the masking threshold, it can be dropped without affecting the perceived quality of the signal. The forward masking threshold is given by

$$m_f(n; n_i, k) = \alpha_k (z_i - C(n_i, k)) (\beta_k - \log_{10}(n))$$  \hspace{1cm} (12)

$$\text{round}(n_i + 0.1l_k + 1) \leq n \leq \text{round}(n_i + 0.1l_k + \rho_k f_s)$$  \hspace{1cm} (13)

where $z_i$ (in dB) is the sensation level of the $i$-th kernel (defined below), $n_i$ is the start time index of the $i$-th kernel ($f_s = 44.1$ kHz), and $\alpha_k$, $\beta_k$, and $l_k$ are given by

$$\alpha_k = \log_{10}(n_i + 0.1l_k + \rho_k f_s) - \log_{10}(n_i + 0.1l_k + 1)$$  \hspace{1cm} (14)

$$\beta_k = \log_{10}(n_i + 0.1l_k + \rho_k f_s)$$  \hspace{1cm} (15)

$$l_k = \text{round}(d_k f_s)$$  \hspace{1cm} (16)

and $C(n, k)$ (in dB) is an offset value in critical band $k$ and time index $n$, subtracted from the sensation level to determine the masking threshold. Our experiments have shown that for a strongly tonal part of the spectrum, this offset is around 20 dB. However, for noise-like parts of the spectrum, this offset
can be reduced to elevate the masking threshold. Using informal listening tests, we have defined the following offset value as a function of the tonality level in each critical band for the frames of 1024 audio sample.

\[ C(n_i, k) = 4 \Gamma(n_i, k) + 16, \]  

(17)

where \( \Gamma(n_i, k) \) is the tonality index for critical band \( k \) at time index \( n_i \). The tonality index is between zero (for noise-type signal) and 1 (for a pure sinusoid). For more details on how the tonality index is computed, the reader is referred to appendix A of [Najaf-Zadeh et al., 2008].

The sensation level \( z_i \) is defined as

\[ z_i = 10 \log_{10} \left( \frac{a_i^2 p_k^2}{q_k} \right), \]  

(18)

where \( a_i \) is the magnitude of the \( i \)-th extracted kernel, \( p_k \) is the peak value of the Fourier transform of the normalized kernel in critical band \( k \), and \( q_k \) is the elevated threshold of hearing in quiet for the same critical band.

Since the effective duration of gammatone kernels are less than 200 ms, the absolute threshold of hearing is elevated by 10 dB/decade [Zwicker and Fastl, 1990]. The elevated threshold in quiet in critical band \( k \) is given by

\[ q_k = \Upsilon_k + 10 (\log_{10}(200) - \log_{10}(d_k)), \]  

(19)

where \( \Upsilon_k \) is the absolute threshold of hearing in critical band \( k \).

The induced forward masking contributes to the global masking pattern in critical band \( k \) and is updated as follows:

\[ \max(M_{TOT}(n_i + 0.1 l_k + 1 : n_i + 0.1 l_k + \rho_k f_s, k), m_f(n; n_i, k)) \rightarrow M_{TOT}(n_i + 0.1 l_k + 1 : n_i + 0.1 l_k + \rho_k f_s, k) \]  

(20)

where the arrow indicates an update of the variable \( M_{TOT} \).

4.1.2. Backward Masking in the Same Critical Band

Backward masking happens when a maskee starts before a masker. For the backward masking, we assume that \( d_b \), the effective duration of masking for all critical bands regardless of the effective duration of the kernels, is equal to 3 ms. Hence, the backward masking threshold is given by

\[ m_b = \gamma_k (z_i - C(n, k)(\log_{10}(n) - \eta_k)) \]  

(21)

\[ n_i - d_b f_i \leq n \leq n_i - 1, \]  

(22)

where

\[ \gamma_k = \log_{10}(n_i - 1) - \log_{10}(n_i - d_b f_s) \]  

(23)

\[ \eta_k = \log_{10}(n_i - d_b f_s) \]  

(24)
The backward masking caused by the extracted kernel contributes to the global masking pattern as follows:

\[
\max (M_{\text{TOT}}(n_i - d_b f_s : n_i - 1, k), m_b(n; n_i, k)) \rightarrow M_{\text{TOT}}(n_i - d_b f_s : n_i - 1, k)
\]  \hspace{1cm} (25)

4.2. Simultaneous Masking in the Same Critical Band

When a maskee starts within a very small delay (i.e., less than 0.1 of the masker duration) after the masker, the masking threshold is given by:

\[
m_s(n; n_i, k) = z_i - C(n_i, k)
\]  \hspace{1cm} (26)

\[
n_i \leq n \leq n_i + 0.1k
\]  \hspace{1cm} (27)

and the contribution to the global masking pattern is

\[
\max (M_{\text{TOT}}(n, k), m_s(n; n_i, k)) \rightarrow M_{\text{TOT}}(n, k)
\]  \hspace{1cm} (28)

4.3. Masking effect on adjacent critical bands

We have also considered the masking effects caused by any extracted kernel in two adjacent critical bands. According to [Zwicker and Fastl, 1990] a single masker produces an asymmetric linear masking pattern in the Bark domain, with a slope of -27 dB/Bark for the lower frequency side and a level-dependent slope for the upper frequency side. The slope for the upper frequency side is given by

\[
s_u = -24 - \frac{230}{f_m} + 0.2z,
\]  \hspace{1cm} (29)

where \( f_m \) is the masker frequency in Hz and \( z \) is the masker level in dB. We have used this approach to calculate the masking effects caused by each spike in the two immediate neighboring critical bands.

\[
\max (M_{\text{TOT}}(n, k - 1), M_{\text{TOT}}(n, k) - 27) \rightarrow M_{\text{TOT}}(n, k - 1)
\]  \hspace{1cm} (30)

\[
\max (M_{\text{TOT}}(n, k + 1), M_{\text{TOT}}(n, k) - s_u) \rightarrow M_{\text{TOT}}(n, k + 1)
\]  \hspace{1cm} (31)

4.4. Perception-Based Signal Decomposition

In matching pursuit, at each iteration the value and position of the maximum of the cross correlation of the residual signal and each kernel is found. The kernel with the highest absolute correlation is selected. The maximum absolute value of the cross correlation and its position are determined. In our method (i.e., Perceptual Matching Pursuit) prior to finding the maximum absolute value for each correlation vector, the absolute values below the global masking surface are set to zero. In other words, the correlation at any time index is taken into consideration if its sensation level is above the associated masking threshold at that time index,

\[
\frac{a_k^2(n) p_k^2}{q_k} > 10^{M_{\text{TOT}}(n, k)/10},
\]  \hspace{1cm} (32)

14
\[ |a_k(n)| > \frac{\sqrt{q_k 10^{\text{MTOT}(n,k)/10}}}{p_k} \tag{33} \]

As such, only audible kernels are extracted and the masked values in the correlation sequences will be discarded. By doing so, the noise spectrum (i.e., residual spectrum) is shaped and a higher noise level is allowed as long as it is inaudible. Fig. 9 shows the power spectrum of a frame of an audio signal and also the spectra for the residual for the matching pursuit and PMP algorithms. As is seen, the PMP algorithm shapes the noise spectrum and therefore produces higher quality audio for the same number of extracted kernels. Fig. 10 shows the masking pattern produced by a spike (with \( f_c = 150\text{Hz} \)) positioned at \( t_0 \). Fig. 10 shows that the time span of the backward masking is much shorter than the time span of forward masking.

![Figure 9: Power spectrum of the residual error in PMP and MP. The power spectrum of the original audio signal is also plotted for comparison purposes.](image)

![Figure 10: Masking pattern produced by a spike (with \( f_c = 150\text{Hz} \)) positioned at \( t_0 \).](image)
4.5. **Subjective Test results**

In order to verify the objective scores, we conducted a semi-formal listening test, based on the ITU.R BS. 1116 method, to evaluate the quality of the test signals. Six subjects took part in a triple stimulus hidden reference test and listened to the audio materials (presented in Table 3) over the headphone in a quiet room. The CRC SEAQ software was used in the test which allowed the listener to seamlessly switch among the three stimuli. In each trial, the stimulus A was always the reference stimulus known by the subject. Two other stimuli, B or C, were either a hidden reference, identical to A, or a synthesized version of the same audio material. None of B or C was known to the subject. The listener had to identify the synthesized version (either B or C) and to grade its quality relative to that of the reference on A. The grading scale was continuous from 1 (very annoying) to 5 (no difference between the reference and the synthesized file). During the listening test, each subject was free to take as much time as required on any trial, switching freely among the three stimuli as often as desired. The average subjective scores for MP and PMP were 3.994 and 4.331, and the standard deviations of the scores were 0.299 and 0.202 respectively. Values in Table 3 are the mapping of the subjective test scores (between 1 and 5) to the Objective Difference Grade (ODG) that varies between -4 to +4, according to the ITU.R BS 1116 standard. Positive values in the ODG represents evaluation errors by subjects (basically errors in identifying the hidden reference), while negative values are the subjective scores (on the 1 to 5 scale) minus 5 in the case where no error in identifying the hidden reference is made by the subject. Although the confidence intervals for the subjective scores are overlapping, all the test materials received higher subjective scores for PMP, which is consistent with the objective evaluation. Results in Table 3 are obtained by activating only the “Matching Pursuit”, “Spikegram Generation”, and “Time-Frequency Plane Masking” blocks of Fig. 1 to assess the effectiveness of our proposed PMP (the “Time-Frequency Plane Masking” block is off for MP).

5. **Frequent Episodes in Spikes**

As mentioned earlier, in the spikegrams generated in section 3.1, the spike activity of each channel can be associated to the activity of a neuron tuned to the center frequency of that channel. The ultimate goal in the pattern recognition paradigm is to find a generative neural architecture (such as a synfire chain [Abeles, 1991] or a polychronous network [Izhikevich, 2006]) that is able to generate a spikegram such as the one we extract by MP (see Fig. 2) for a given audio signal. In this article, we propose a solution to a simplified version of the aforementioned problem. We propose to extract “channel-based or frequency-domain patterns” in our generated spikegrams using temporal data mining [Mannila et al., 1997] [Patnaik et al., 2008]. Since these patterns are

---

2http://www.itu.int/rec/R-REC-BS.1116-1-199710-I/e
Table 3: Mean subjective and objective scores for a few audio files processed with MP and PMP. Objective Difference Grade (ODG) are shown in the table for subjective tests. The spike rate is chosen in such a way that either MP or PMP achieves an objective score better than -1 (i.e., broadcast quality).

5.1. Frequent Episode Discovery

Frequent Episode Discovery framework was proposed by Mannila et al. [Mannila et al., 1997] and enhanced in [Laxman et al., 2007]. Patnaik et al. [Patnaik et al., 2008] extended previous results to the processing of neurophysiological data. The frequent episode discovery fits in the general paradigm of temporal data mining. The method can be applied to either serial episodes (ordered set of events) or to parallel episodes (unordered set of events). A frequent episode is one whose frequency exceeds a user specified threshold. Given an episode occurrence, we call the largest time difference between any two events constituting the occurrence as the span of the occurrence and we use this span as a temporal constraint in the algorithm. The overall procedure for frequent episode discovery is presented in Table 4 as a pseudo code.

In frequent episode discovery framework, the data to be analyzed is a sequence of events denoted by \((E_1, t_1), (E_2, t_2), (E_3, t_3)\) where \(E_i\) represents an event type and \(t_i\) the time of occurrence of the \(i\)-th event. \(E_i\) are drawn from a finite set of event types. The sequence is ordered with respect to the time of occurrences of events so that \(t_i \leq t_{i+1}\) for all \(i = 1, 2, \ldots\) The following is an example event sequence containing 6 events with 4 event types:

\[\langle (A, 1), (B, 3), (D, 4), (C, 6), (A, 12), (B, 15) \rangle\]

In frequent episode discovery the goal is to detect 2 types of sequences. Serial episodes that are ordered tuples of events. For example \((A \rightarrow B \rightarrow C)\) is a 3-node serial episode. The arrows in this notation indicate the order of the event types in time.

A parallel episode is an episode in which the order of occurrence does not have any importance. An occurrence \((ABC)\) can have the events in any order in the sequence. In this paper, our focus is on parallel episode discovery.
The reader may note that these sequence of events could not be extracted by conventional pattern recognition techniques since the time elapsed between two events can be very different from one episode to another in temporal data mining and still be considered as similar episodes. For instance, sequences \( ((A, 1), (B, 3), (C, 4)) \) and \( ((A, 1), (B, 10), (C, 14)) \) are considered different patterns in a classical pattern recognition system, while they are two similar serial episodes. Furthermore, sequences \( ((A, 1), (B, 3), (D, 4)) \) and \( ((B, 2), (C, 10), (A, 14)) \) are considered as being similar parallel episodes.

<table>
<thead>
<tr>
<th>Generate an initial set of (1-node) candidate episodes ((N = 1))</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>repeat</strong></td>
</tr>
<tr>
<td>Count the number of occurrences of the set of (N-node) candidate episodes</td>
</tr>
<tr>
<td>Retain only those episodes whose count is greater than the frequency threshold (frequent episodes)</td>
</tr>
<tr>
<td>Using the set of (N-node) frequent episodes, generate the next (N+1-node) candidate episodes</td>
</tr>
<tr>
<td><strong>until</strong> There are no candidate episodes remaining</td>
</tr>
<tr>
<td>Output all the frequent episodes discovered</td>
</tr>
</tbody>
</table>

Table 4: The frequent episode discovery algorithm as described in [Patnaik et al., 2008].

5.2. Extraction of frequency-domain patterns in spikegrams

Given the sequence of spike channel number \((f_i, f_k, \ldots, f_m)\) where \(i, k, m\) vary between 1 and \(N\), the number of channels (associated with centre frequencies) in the spikegram, we want to find frequent parallel episodes that are subsets of the sequence given above. The frequent episodes represent the underlying statistical dependencies between different center frequencies for a given time interval specified by the temporal constraint of the discovery algorithm. The frequent episodes here can be considered as “frequency-based audio objects” since they are the frequency-domain building blocks of our audio signal. In graphical terms, frequent episodes are visual structures that repeat frequently on the spikegram within a predefined time window. Since we are looking for unordered episodes, the aforementioned structures are similar up to a permutation in the order of appearance. This can be roughly compared to extracting similar regions on a conventional spectrogram. However, the spikegram is much more precise than a spectrogram in terms of the ability in extracting acoustic events (or timing information). In addition, the spikegram can only take on discrete values. Hence, it is much easier to extract patterns in such a discrete representations compared to a spectrogram where values are continuous. As we will see in section 5.3, the sequence \((f_i, f_k, \ldots, f_m)\) can be represented in terms of frequent episodes (that we will use as elements of a codebook) plus the residual center frequency sequence (that cannot be expressed in terms of codebook elements). We only consider patterns (codebook elements) for which their length multiplied by their frequency of repetition is higher than a predefined threshold. In addition, we noticed that spikegrams are denser in some regions than others. Therefore, the pattern extraction will be normally biased towards those regions and sparser regions will be ignored when the pattern extraction
Table 5: Results for a 3-Pass pattern extraction on 1-second frames. **Percussion:** The total number of bits to address channels when no pattern recognition is used equals 23704 and the saving in addressing channels due to our algorithm is 49%. **Castanet:** The total number of bits to address channels when no pattern recognition is used is 21982 and there is a saving of 26% with our proposed algorithm. **Speech:** The total number of bits to address channels when no pattern recognition is used is 19118 and there is a saving of 40%.

<table>
<thead>
<tr>
<th></th>
<th>Percussion</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pass 1</td>
<td>Pass 2</td>
<td>Pass 3</td>
<td>Overall</td>
</tr>
<tr>
<td>No. extracted spikes</td>
<td>1682</td>
<td>771</td>
<td>335</td>
<td>2788</td>
</tr>
<tr>
<td>No. codebook elements</td>
<td>47</td>
<td>36</td>
<td>11</td>
<td>94</td>
</tr>
<tr>
<td>Codebook size in bits</td>
<td>2200</td>
<td>1976</td>
<td>320</td>
<td>4496</td>
</tr>
<tr>
<td>Raw bit saving</td>
<td>9968</td>
<td>4403</td>
<td>1820</td>
<td>16191</td>
</tr>
<tr>
<td>Effective bit saving</td>
<td>7768</td>
<td>2427</td>
<td>1500</td>
<td>11695</td>
</tr>
<tr>
<td></td>
<td>Castanet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pass 1</td>
<td>Pass 2</td>
<td>Pass 3</td>
<td>Overall</td>
</tr>
<tr>
<td>No. extracted spikes</td>
<td>596</td>
<td>684</td>
<td>580</td>
<td>1860</td>
</tr>
<tr>
<td>No. codebook elements</td>
<td>8</td>
<td>20</td>
<td>37</td>
<td>65</td>
</tr>
<tr>
<td>Codebook size in bits</td>
<td>440</td>
<td>1436</td>
<td>2340</td>
<td>4216</td>
</tr>
<tr>
<td>Raw bit saving</td>
<td>2660</td>
<td>4095</td>
<td>3253</td>
<td>10008</td>
</tr>
<tr>
<td>Effective bit saving</td>
<td>2220</td>
<td>2659</td>
<td>913</td>
<td>5792</td>
</tr>
<tr>
<td></td>
<td>Speech</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pass 1</td>
<td>Pass 2</td>
<td>Pass 3</td>
<td>Overall</td>
</tr>
<tr>
<td>No. extracted spikes</td>
<td>1262</td>
<td>689</td>
<td>395</td>
<td>2346</td>
</tr>
<tr>
<td>No. codebook elements</td>
<td>8</td>
<td>21</td>
<td>11</td>
<td>40</td>
</tr>
<tr>
<td>Codebook size in bits</td>
<td>338</td>
<td>1053</td>
<td>288</td>
<td>1679</td>
</tr>
<tr>
<td>Raw bit saving</td>
<td>3238</td>
<td>3859</td>
<td>2250</td>
<td>11026</td>
</tr>
<tr>
<td>Effective bit saving</td>
<td>2899</td>
<td>2806</td>
<td>1962</td>
<td>7667</td>
</tr>
</tbody>
</table>

algorithm is applied just once. Hence, we propose a multipass approach in which patterns are extracted during the first pass in denser regions. We then subtract the patterns we matched to the spikegram from the spikegram and we keep the residual spikegram on which we run the frequent episode discovery algorithm a second time. Finally, we apply the frequent episode discovery algorithm on the residual spikegram of the second pass. Our observations have shown that very little information is extracted after the third pass. Therefore, we use a 3-pass approach throughout this article. The GMiner toolbox\[^{3}\]\[^{[Patnaik et al., 2008]}\] based on the pseudo-code in Table 4 is used to extract patterns in our spikegrams.

5.3. Pattern Discovery and Coding Results

In this section we give pattern discovery results for three different audio signals: percussion, castanet, and speech.

\[^{3}\]http://neural-code.cs.vt.edu/
5.3.1. Experimental Setup

The signal is processed in 1 second frames. For each frame, a 4000-spike spikegram is generated. Frequent episodes are discovered for each signal during three different passes as described in section 5.2. The temporal constraint window is set to 400 (at a 44.1kHz rate), meaning that the occurrence time of any two spikes in an episode cannot exceed 400 discrete samples. The minimum threshold for the frequency of episode multiplied by the length of the episode is set to 10. Therefore, very short sequences or rarely-occurring sequences are not extracted, as they do not result in significant bit saving. Each element of the codebook is run-length coded and sent to the receiver. The total number of bits required to send the codebook to the receiver is computed as well. For each pass the residual spikes are arithmetic coded and the difference in the number of bits required to code the residual at each pass is computed as “raw bit saving” in
addressing channels. We then computed the “effective bit saving” in addressing channels as the “raw bit saving” minus the bits required to send the codebook (overhead). This is the effective gain obtained in bitrate when our proposed 3-pass pattern extraction is used (see Table 5).

Figure 12: Residual norm (\(|r_x(t)|\) in Eq. 1) vs. number of iterations for percussion when 24 and 64 channels are used for spike extraction.

5.3.2. Coding Results in Terms of Bitrate

In Table 5, the number of extracted spikes is shown for each pass and the raw bit saving and effective bit saving in addressing channels as described above are given for percussion, castanet, and speech. Our algorithm was able to extract between 1860 and 2788 spikes in different episodes out of the total 4000 spikes. The longest pattern found in percussion is 13-spike long and is repeated on average 17 times in the signal frame, while the longest pattern for castanet is 14-spike long and is repeated 33 times on average in frames. In the meantime, the longest pattern for speech is 100-spike element and is repeated 8 times on average in the frames. Results show that the bitrate coding gain obtained in addressing frequency channels ranges from 26% to 49% depending on signal type. Note that since the pattern extraction coding is lossless, the semi-formal subjective quality evaluations in [Pichevar et al., 2007] for the audio materials still hold when our new audio extraction paradigm is applied. Fig. 11 shows the extracted patterns for each of the three distinct passes for percussion. Since unordered episodes are discovered, the order of appearance of spikes in different channels can change. However, the channels in which spike activity occurs are the same for all similar patterns. Fig. 11 also shows that our 3-pass algorithm is able to extract patterns in the high, low and mid-frequency ranges, while a 1-pass algorithm would have penalized some sparser spikegram regions.

5.4. Extracted Patterns in Spectro-Temporal Domains

Fig. 13 shows how the precise timing of a percussion signal can be represented by a few codebook elements. For instance, reconstruction with the first codebook element extracted by our proposed algorithm (13-spike long and repeated 17 times in the signal) shows that with only this first element a considerable amount of the signal is grabbed at each energy burst with accurate timing.
Figure 13: Reconstruction of a percussion signal with a few codebook elements. **First Panel:** Original percussion signal. **Second to Fifth Panels:** Signals generated with the first to fourth codebook elements respectively.

<table>
<thead>
<tr>
<th>Bits/spike</th>
<th>24-channel without PE</th>
<th>64-channel with PE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel</td>
<td>5.6</td>
<td>3.2</td>
</tr>
<tr>
<td>Time</td>
<td>10.1</td>
<td>10.1</td>
</tr>
<tr>
<td>Amplitude</td>
<td>3.9</td>
<td>3.9</td>
</tr>
</tbody>
</table>

Table 6: Average number of bits used to address each parameter in the 24-channel without Pattern Extraction (PE) and the 64-channel with PE cases.

Fig. 14 shows how codebook elements represent frequency-domain information for the same percussion signal. The reader may notice how some frequency-domain patterns (especially on panels 4 and 5 of Fig. 14) are flipped/mirrored versions of each other. For instance, let us consider the two spectral patterns at times $2.2 \times 10^4$ and $2.6 \times 10^4$ on panel 5 of Fig. 14 (as indicated by arrows in the Figure). The reader may notice that in the first spectral pattern, the dark/red zone around 14 kHz precedes the dark/red zone in the mid-level frequency range (8 kHz), while for the pattern located at $2.6 \times 10^4$ the opposite happens and the 8 kHz dark zone precedes the 14 kHz dark zone (indicated by arrows). This flexibility in finding symmetrical (temporally-mirrored) patterns is due to the fact that our algorithm is based on the extraction of parallel frequent episodes (unordered set of events), so that the relative timing of different “high-energy” (dark) zones can change in a pattern. This interesting feature reduces the number of elements in the codebook drastically, since all mirrored patterns are classified as a single codebook element in our algorithm.
5.5. Choice of the Window Size and Maximum Episode Length Parameters

Fig. 15 shows a boxplot the number of occurrences of an episode for a given length. More specifically, the frequency episode discovery algorithm found on average 4000 different 10-spike-long serial patterns (grouped into different classes) per second in a percussion signal (this information is not depicted in Fig. 15). Note that the 10-spike-long patterns are overlapping (some spikes are common between different patterns), which explains why the number of extracted patterns is higher than the number of extracted spikes. According to Fig. 15, the maximum number of repetitions for 10-spike-long episodes was 23, while the minimum was 7. The 25th percentile of counts was 13 while the 75 percentile was 10. This kind of information helps us decide the range of episode lengths that should be used for coding purposes. For instance, for 40-spike-long episodes the maximum repetition in one second is 4. Therefore, the maximum saving we can get is 160 spikes minus 40 spikes (as side information to be transmitted as a codebook element for that specific pattern). Therefore, we decided to not include discovered episodes that give a gain in the number of saved spikes lower than 200, to reduce computational complexity and memory usage.

Fig. 16 shows the effect of the size of the temporal constraint window (the largest time difference between any two events constituting the occurrence) on the maximum length (in terms of spikes) of the discovered episode as well as the average number of repetitions found in 1 second for speech (the type of signal for which the longest episodes are discovered). The graph shows that the optimal
Figure 15: Box plot showing the number of occurrences of episodes for different episode lengths (number of spikes in the episodes) for percussion. For each length, the minimum number (lower-bound whiskers), the first quartile (rectangles’ lower bound), the third quartile (rectangles’ upper bound), and the maximum number (upper-bound whiskers) of occurrences are plotted.

value for the temporal constraint window is 400 (there is neither a change in the number of occurrences nor in the maximum length of discovered episodes beyond this value). Therefore, the temporal constraint window of length 400 is used in our experiments as explained before.

5.5.1. Choice of Number of Channels in The Spikegram

Fig. 12 shows that the number of spikes required to get the same SNR decreases considerably when 64 channels are used instead of 24 in the spikegrams (see [Pichevar et al., 2007] for details). In section 3, we analyzed the impact of the number of channels on the overall number of spikes required for a given audio quality. We also mentioned that this reduction in the number of spikes, due to an increase in the number of channels, comes at the cost of an increase in the bitrate addressing channel information. However, since we propose in this section our pattern discovery algorithm that reduces the need to address channels individually, we can increase the number of channels when our pattern recognition algorithm is used. Table 6 shows the average number of bits required to address each parameter in the cases when pattern extraction is used and when it is not, for 24-channel and 64-channel spikegrams. When 64 channels are used the total number of spikes required for a given SNR (as shown by the horizontal dashed line in Fig. 12) is 2400, while for the same SNR we need 4000 spikes in the 24-channel case (informal listening tests confirm the results). Therefore, the total number of bits used to address time, channel, and amplitude in 24-channel (without pattern extraction) and 64-channel (with pattern extraction) spikegrams are 78400 and 41280 bits respectively (based on the data in Table 6). Thus, there is a saving of 47% in the total number of bits and therefore our choice of using 64-channel spikegrams in the previous section is justified. The reader may notice that the 24-channel with pattern extraction and 64 channel without pattern extraction scenarios are less optimal in terms of bitrate saving
and are not considered in Table 6.

6. Conclusion and future work

We have proposed a biologically-inspired paradigm for universal audio coding based on neural spikes. Our proposed approach is based on the generation, by matching pursuit, of auditory-inspired sparse 2-D representations of audio signals, dubbed as spikegrams. We showed that the use of an adaptive scheme that dynamically changes the tuning parameters in the gammachirp kernels considerably decreases spike counts and the associated bitrate to code audio materials for a given quality when compared to the non-adaptive paradigm. We then applied simultaneous, temporal backward, and temporal forward masking models to further reduce spike counts for a given perceived quality and from there proposed the Perceptual Matching Pursuit that defines an automatic stop criterion based on the masking threshold.

We finally proposed a fast (faster than the MP stage) frequency-domain audio object (episode) extraction algorithm based on the generation of spikegrams. The advantage of such an algorithm stems in the fact that spikegrams are representations of discrete events that can be mined easily by known approaches. This is in contrast with raw or frequency-domain representations of signals (i.e., spectrogram) in which each sample can take so many values and where data mining is difficult to perform. We then applied our proposed technique to audio coding and obtained promising results for the lossless coding of frequency-based information. In order to increase performance, we proposed a 3-pass pattern extraction method that helps extract patterns more uniformly in spikegrams. The advantage of our proposed method is two-fold. First, we show how to save bits by extracting patterns and small codebooks for sending channel informa-
tion with a much lower bitrate. We also obtained another bitrate decrease due to the fact that by increasing the number of channels in the spikegram, we can decrease the number of spikes needed to meet the same quality. This aforementioned gain is achieved due to the efficiency in sending channel information collectively as patterns. Semi-formal listening tests show that the overall system in Fig. 1 gives high quality (scores above 4 on the ITU-R 5-grade impairment scale) and has the potential to achieve the target 44.1 kbps for the audio material described in this article. We will also investigate the relation between the statistics for the adaptation parameters $l$, $b$, and $c$ with the type of the signal for which the representation is extracted. This could potentially lead to a way to classify signals.

In a future work, we will extract the structural dependencies of spike amplitudes and/or other parameters in the spikegram such as the chirp factor, etc. We will also investigate the design of a generative neural model based on spikegrams. Formal subjective listening tests for the overall system will be conducted in the future. In order to speed up the spikegram extraction of audio signals, we have conducted tests on replacing the MP stage (see Fig. 1) by neural circuitry that can be implemented on embedded and parallel hardware [Rozell et al., 2008] [Pichevar et al., 2010]. We will further explore this avenue in a future work. The application of different ideas outlined in this article (i.e., pattern recognition and masking model) are not limited to spikegrams and can be applied to other sparse representations in the literature (i.e., [Ravelli et al., 2008] [Abdallah and Plumbley, 2006]). In addition, the frequency episode discovery algorithm discussed in this article can be used in speech recognition, sound source separation, and audio classification.

Acknowledgments

The authors would like to thank Richard Boudreau, Hunter Hong, and Frederic Mustiere for proofreading the paper. They also express their gratitude to Debrprakash Patnaik and Koniparambil Unnikrishnan for providing them with the GMiner toolbox and for fruitful discussions on frequent episode discovery. The first author would also like to thank Jean Rouat for fruitful discussions on machine learning, as well as the University of Sherbrooke for a travel grant that made the discussions on machine learning and frequent episode discovery possible. Many thanks also to the three anonymous reviewers for their constructive comments.

References


