PATTERN EXTRACTION IN SPARSE REPRESENTATIONS WITH APPLICATION TO AUDIO CODING

Ramin Pichevar and Hossein Najaf-Zadeh

Communications Research Centre, 3701 Carling Ave., Ottawa, Canada
Ramin.Pishehvar@crc.ca, Hossein.NajafZadeh@crc.ca

1. ABSTRACT

This article deals with the extraction of frequency-domain auditory objects in sparse representations. To do so, we first generate sparse audio representations we call spikegrams, based on neural spikes using gammatone/gammachirp kernels and matching pursuit. We then propose a method to extract frequent auditory objects (patterns) in the aforementioned sparse representations. The extracted frequency-domain patterns help us address spikes (atoms or auditory 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. Our experiments show that substantial coding gain is obtained when our technique based on pattern extraction is used as opposed to the case where spikes (atoms) are coded individually. This work is a first step towards the design of a high-quality "object-based" audio coder.

2. INTRODUCTION

In [9], we proposed a bio-inspired universal audio coder based on projecting signals onto a set of overcomplete atoms consisting of gammatone/gammachirp kernels (see [12] for a literature survey on overcomplete sparse audio coding). The projections on the kernels are called spikes, since they can be considered as the spikes generated by hair cells in the auditory pathway (see Fig. 1). The best atom at each iteration is found by matching pursuit. Our proposed method in [9] is an adaptive version of [14] and uses gammachirp kernels instead of the original gammatones used in [14]. In our approach, at each matching pursuit iteration, six different parameters (i.e., amplitude, time, frequency, chirp factor, attack, and decay) are extracted in the adaptive case, while three parameters (i.e., amplitude, time, frequency) are extracted in the non-adaptive case. Note that our approach is different from other works in the literature (i.e., [3]) in which gammatones are used as a filterbank and not as kernels for the generation of sparse representations based on matching pursuit. We also showed in [9] that, when used for audio coding, our adaptive approach outperforms the work in [14] in terms of bitrate and number of atoms for the same perceptual quality on different types of signals. The representations we dubbed as spikegrams are good at extracting non-stationary and time-relative structures such as transients, timing relations among acoustic events, and harmonics.

The authors would like to thank D. Patnaik and K. Unnikrishnan for the GMiner toolbox and for fruitful discussions, J. Rouat for fruitful discussions, as well as the University of Sherbrooke for a travel grant that made the aforementioned discussions possible.

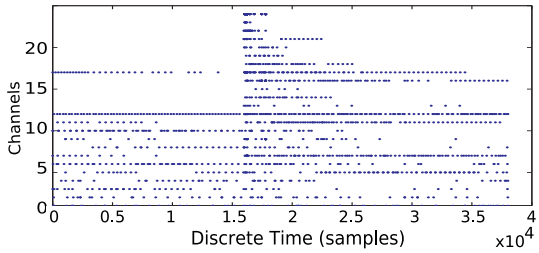

Figure 1: Spikegram of the harpsichord using the gammatone MP algorithm (spike amplitudes are not represented). Each dot represents the time and the channel where a spike is fired.

In [9], we only studied the analysis/synthesis of a given signal using our proposed method when each spike is processed individually and when the underlying signal is synthesized as the sum of all individual spikes. The aforementioned approach lacks the ability to process auditory information in holistic form (i.e., as auditory objects [2]) and therefore encodes each spike (auditory event) individually. Hence, the statistical dependence between spikes/atoms that form auditory objects is not exploited specifically in [9] and more generally in other sparse representations in the literature. In this article, we propose an approach that takes into consideration the statistical dependence between some spike attributes and is therefore a more optimal way to represent auditory signals.

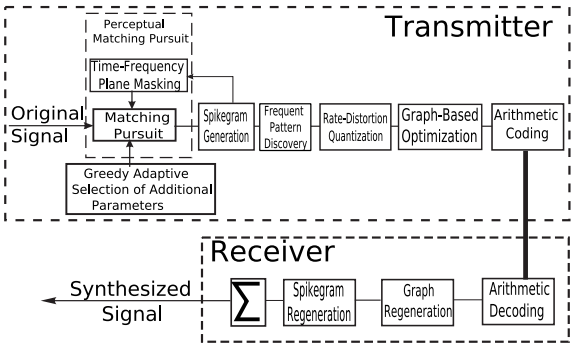

Figure 2: Block diagram of our proposed Universal Bio-Inspired Audio Coder.

3. THE BIO-INSPIRED AUDIO CODER

The analysis/synthesis part of our universal audio codec presented in [9] is based on the generation of sparse 2-D rep-

Generate an initial set of (1-node) candidate episodes ($N = 1$)
repeat
Count the number of occurrences of the set of (N-node) candidate episodes in one pass of the data sequence
Retain only those episodes whose count is greater than the frequency threshold and declare them to be frequent episodes
Using the set of (N-node) frequent episodes, generate the next set of (N+1-node) candidate episodes
until There are no candidate episodes remaining
Output all the frequent episodes discovered

Table 1: The frequent episode discovery algorithm as described in [8].

resentations 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 3.1). The adaptiveness is a key feature we introduced in Matching Pursuit (MP) to increase the efficiency of the proposed method (see [9]). A masking model is applied to the spikegrams to remove inaudible spikes [7]. In addition a differential encoder of spike parameters based on graph theory is proposed in [10]. The quantization of the spikes is given in our previous work [11]. The block diagram of all the building blocks of the receiver and transmitter of our proposed universal audio coder is depicted in Fig. 2, of which the frequent pattern discovery block is discussed in this paper.

3.1 Generation of Overcomplete Representations with MP

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

$$x(t) = < x(t), g_m > g_m + r_x(t), \quad (1)$$

where $< x(t), g_m >$ is the inner product between the signal and the kernel g_m. $r_x(t)$ is the residual signal after projection. In order to find an adequate representation as in Eq. 1, MP can be used. 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 bases g_m. In [9], we used adaptive gammachirp kernels $g_m(t)$. In the aforementioned approach, the chirp factor (instantaneous frequency), the attack, and the decay of gammachirp kernels are found adaptively in addition to the standard parameters of the gammatone kernels.

In the remainder of this article, a new solution to the extraction of frequent episodes (auditory objects or patterns) out of the generated spikegrams is presented. Without loss of generality, we use the non-adaptive (3-parameter gammatone kernels as $g_m(t)$) approach in this article, since we only extract frequency-domain patterns.

4. FREQUENT EPISODES IN SPIKES

In spikegrams, 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 here is to find a generative neural architecture (such as a synfire chain [1] or a polychronous network [4]) that is able to generate a spikegram such as the one we extract by MP (see Fig. 1) for a given audio signal. Here, 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 [6] [8]. Since these patterns are repeated frequently in the signal and are the building blocks of the audio signal, we may call them auditory objects. Note that spikes' timing and amplitude information is encoded independently as in [9] and is not taken into account in extracted patterns.

Frequent Episode Discovery framework was proposed by Mannila et al. [6] and enhanced in [5]. Patnaik et al. [8] 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 episode discovery is presented in Table 1 as a pseudo code.

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

Table 2: 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% (compared to when no pattern discovery is used as in [9]). **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%.

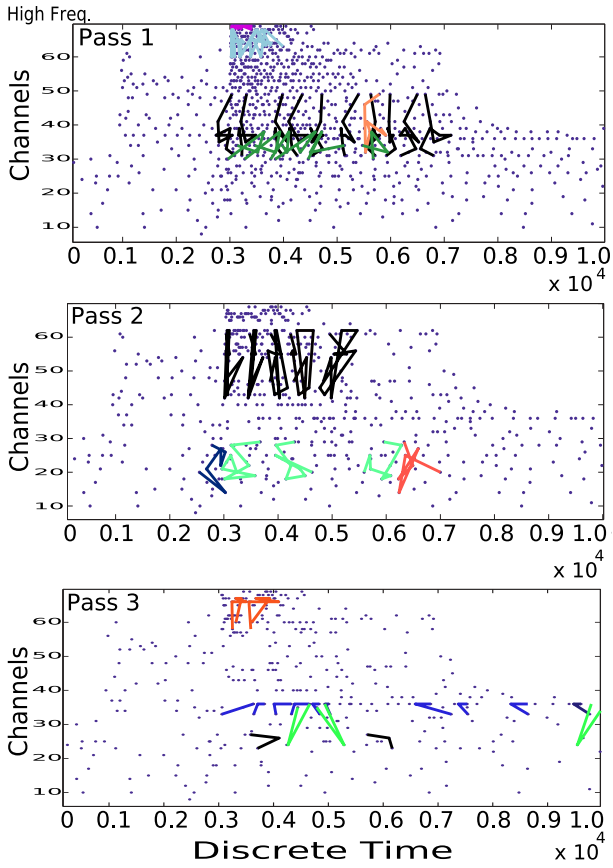

Figure 3: Spikegrams (dots) and the most *relevant* extracted patterns (lines) at each of the 3 passes for percussion for a 250 ms frame. Different colors/grayscales represent different episodes. Only spikes not discovered during the previous pass are depicted at each pass. Note that since unordered episodes are discovered, patterns are similar up to a permutation in the temporal order. Timing and amplitude information is not included in the patterns and is encoded separately.

4.1 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 auditory objects" since they are the frequency-domain building blocks of our audio signal and they do not include timing or amplitude information (timing and amplitude are sent individually and independently from the patterns). 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, in contrast with spectrograms, spikegrams are reversible (e.g., one can synthesize the original signal from spikegram elements) . In addition, the spikegram is much more precise than a spectrogram in terms of the ability in extracting acoustic events (or timing information). Furthermore, 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 the section 5, the sequence $(f_i, f_k, \ldots, f_m)$ can be expressed 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. Note that other parameters such as spikes' timing and amplitude are encoded separately as in [9]. We only consider patterns (codebook elements) for which their length multiplied by their number of occurrence is higher than a predefined threshold. Furthermore, we noticed that spikegrams are denser in some regions than others. Therefore, the extraction of patterns would be normally biased towards those regions and sparser regions would be ignored, if the pattern extraction algorithm was 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[1][8] based on the pseudo-code in Table 1 is used to extract patterns in our spikegrams. The input to the GMiner toolbox at each pass is either the original spikegram (first pass) or the residual spikegram (passes 2 and 3) as described above.

5. RESULTS

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

5.1 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 4.1. The temporal constraint window is set to 400, meaning that the difference of occurrence time of any two spikes in an episode cannot exceed 400 discrete samples. The threshold (i.e., number of episode occurrence 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 channel addressing . We then computed the "effective bit saving" in channel addressing as the "raw bit saving" minus the bits required to send the codebook (overhead). This is the effective gain obtained in

[1] http://neural-code.cs.vt.edu/

bitrate when our proposed 3-pass pattern extraction is used (see Table 2).

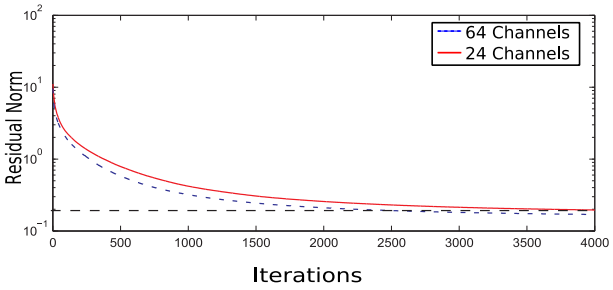

Figure 4: Residual norm ($\|r_x(t)\|$ in Eq. 1) vs. number of iterations for percussion when 24 and 64 channels are used for spike extraction. Each iteretion is associated with a spike.

5.2 Pattern Discovery and Coding Results

In Table 2 the number of extracted spikes is shown for each pass and the raw bit saving and effective bit saving in channel addressing 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 of 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 the type of the signal. Note that since the pattern extraction coding is lossless, the informal subjective quality evaluations in [9] for the audio materials still hold when our new audio extraction paradigm is applied. Fig. 3 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. 3 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.

5.3 Extracted Patterns in Spectro-Temporal Domains

Fig. 5 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. Fig. 6 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. 6) are flipped/mirrored versions of each other. For instance, let us consider the two spectral patterns at times 2.2×10^4 and 2.6×10^4 on panel 5 of Fig. 6 (as indicated by arrows in the Figure). The reader may notice that in the

Bits/spike	24-channel without PE	64-channel with PE
Channel	5.6	3.2
Time	10.1	10.1
Amplitude	3.9	3.9

Table 3: Average number of bits used to address each parameter in the 24-channel without Pattern Extraction (PE) and the 64-channel with PE cases. See [9] for values associated with time and amplitude.

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 2.6×10^4 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.

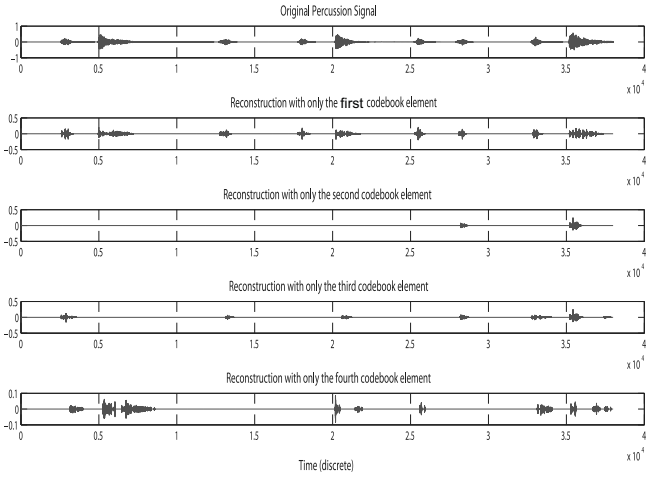

Figure 5: Reconstruction of a percussion signal with a few codebook elements. **1st Panel:** Original percussion signal. **2nd to 5th Panels:** Signals generated with the first to fourth codebook elements respectively.

5.4 Choice of Number of Channels in The Spikegram

Fig. 4 shows that the number of spikes required to get the same SNR decreases drastically when 64 channels are used instead of 24 in the spikegrams. Nevertheless, since a 64-channel spikegram would have required much more bits to address channels individually, in [9] we used the 24-channel spikegram to code spikes individually. However, in the current work, since patterns (i.e., groups of spikes) are extracted, the number of bits required to collectively address channel information is drastically reduced. As such, here we use 64-channel spikegrams. Table 3 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

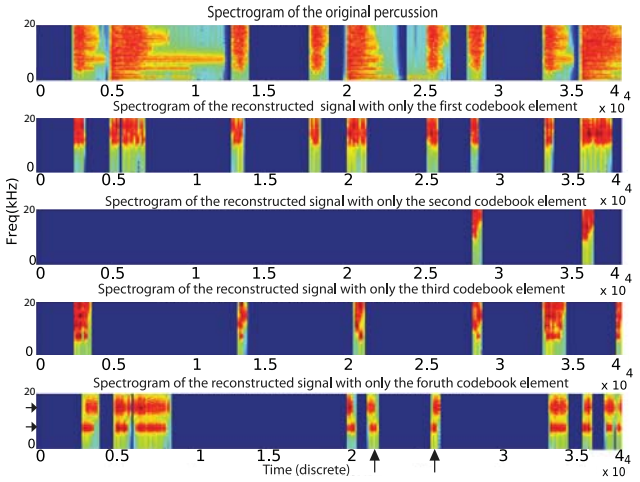

Figure 6: **1st Panel:** Spectrogram of the original percussion signal. **2nd to 5th Panels:** Spectrograms of signals generated with the first to fourth codebook elements respectively.

total number of spikes required for a given SNR (shown by the horizontal dashed line in Fig. 4) is 2400, while for the same SNR we need 4000 spikes in the 24-channel case (confirmed by informal listening tests). 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 3). Thus, there is a saving of 47% in the total bitrate and our choice of using 64-channel spikegrams in the previous sections is justified.

6. CONCLUSION AND FUTURE WORK

We propose 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 irreversible 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 pattern extraction approach is two-fold. First, we show how to save bits by extracting patterns and small codebooks for sending channel information 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. Informal listening tests show that the overall system in Fig. 2 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. In a future work, we will extract the structural dependencies of spike amplitudes, timings, and/or other parameters in the spikegram such as the chirp factor, etc. (see [9]). 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 order to speed up the spikegram extraction of audio signals, we have conducted preliminary tests on replacing the MP stage (see Fig. 2) by neural circuitry that can be implemented on embedded and parallel hardware [13]. We will further explore this avenue in a future work. The application of our proposed audio object extraction is not limited to audio coding and can be used in audio source separation, speech recognition, etc. It can also be applied to sparse representations other than spikegrams.

REFERENCES

[1] M. Abeles. *Corticonics: Neural circuits of the cerebral cortex*. Cambridge University Press, 1991.

[2] A. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT Press, 1994.

[3] C. Feldbauer, G.Kubin, and W.B.Kleijn. Anthropomorphic coding of speech and audio: A model inversion approach. *EURASIP-JASP*, 9:1334–1349, 2005.

[4] E.M. Izhikevich. Polychronization: Computation with spikes. *Neural Computation*, 18:245–282, 2006.

[5] S. Laxman, P. Sastry, and K. Unnikrishnan. Discovery of frequent generalized episodes when events persist for different durations. *IEEE Trans. on Knowledge and Data Eng.*, 19:1188–1201, 2007.

[6] H. Mannila, H. Toivonen, and A. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.

[7] H. Najaf-Zadeh, R. Pichevar, L. Thibault, and H. Lahdili. Perceptual matching pursuit for audio coding. In *Audio Eng. Society Conv., Netherlands*, 2008.

[8] D. Patnaik, P. Sastry, and K. Unnikrishnan. Inferring neural network connectivity from spike data: A temporal mining approach. *Scientific Programming*, 16:49–77, 2008.

[9] R. Pichevar, H. Najaf-Zadeh, and L. Thibault. A biologically-inspired low-bit-rate universal audio coder. In *Audio Eng. Society Conv., Austria*, 2007.

[10] R. Pichevar, H. Najaf-Zadeh, L. Thibault, and H. Lahdili. Differential graph-based coding of spikes in a biologically-inspired universal audio coder. In *Audio Eng. Society Conv., Netherlands*, 2008.

[11] R. Pichevar, H. Najaf-Zadeh, L. Thibault, and H. Lahdili. Entropy-constrained spike modulus quantization in a bio-inspired universal audio coder. In *European Signal Proc. Conf., Lausanne, Switzerland*, 2008.

[12] E. Ravelli, G. Richard, and L. Daudet. Union of MDCT bases for audio coding. *IEEE Transactions on Audio, Speech and Language*, 16(8):1361–1372, 2008.

[13] C. Rozell, D. Johnson, D. Baraniuk, and B. Olshausen. Sparse coding via thresholding and local competition in neural circuits. *Neural Computation*, 20(10):2526–2563, October 2008.

[14] E. Smith and M. Lewicki. Efficient auditory coding. *Nature*, 7079:978–982, 2006.