Sound retrieval and ranking using sparse auditory representations

Richard F. Lyon[1,*], **Martin Rehn**[1], **Samy Bengio**[1], **Thomas C. Walters**[1], **Gal Chechik**[1,*]

[1] Google, 1600 Amphitheatre Parkway, Mountain View CA, 94043

[*] for correspondence: dicklyon@google.com, gal@google.com

Keywords: Auditory processing, sparse code, machine hearing

Abstract

To create systems that understand the sounds that humans are exposed to in everyday life, we need to represent sounds with features that can discriminate among many different sound classes. Here, we use a sound-ranking framework to quantitatively evaluate such representations in a large scale task. We have adapted a machine-vision method, the "passive-aggressive model for image retrieval" (PAMIR), which efficiently learns a linear mapping from a very large sparse feature space to a large query-term space. Using this approach we compare different auditory front ends and different ways of extracting sparse features from high-dimensional auditory images. We tested auditory models that use adaptive pole–zero filter cascade (PZFC) auditory filterbank and sparse-code feature extraction from stabilized auditory images via multiple vector quantizers. In addition to auditory image models, we also compare a family of more conventional Mel-Frequency Cepstral Coefficient (MFCC) front ends. The experimental results show a significant advantage for the auditory models over vector-quantized MFCCs. Ranking thousands of sound files with a query vocabulary of thousands of words, the best precision at top-1 was 73% and the average precision was 35%, reflecting a 18% improvement over the best competing MFCC frontend.

1 Introduction

Machine Hearing is a field aiming to develop systems that can process, identify and classify the full set of sounds that people are exposed to. Like machine vision, machine hearing involves multiple problems: from auditory scene analysis, through "auditory object" recognition to speech processing and recognition. While considerable effort has been devoted to speech and music related research, the wide range of sounds that people – and machines – may encounter in their everyday life has been far less studied. Such sounds cover a wide variety of objects, actions, events, and communications: from natural ambient sounds, through animal and human vocalizations, to artificial sounds that are abundant in today's environment.

Building an artificial system that processes and classifies many types of sounds poses two major challenges. First, we need to develop efficient algorithms that can learn to classify or rank a large set of different sound categories. Recent developments in machine learning, and particularly progress in large scale methods (Bottou et al., 2007), provide several efficient algorithms for this task. Second, and sometimes more challenging, we need to develop a representation of sounds that captures the full range of auditory features that humans use to discriminate and identify different sounds, so that machines have a chance to do so as well. Unfortunately, our current understanding of how the plethora of naturally encountered sounds should be represented is still very limited.

To evaluate and compare auditory representations, we use a real-world task of content-based ranking sound documents given text queries. In this application, a user enters a textual search query, and in response is presented with an ordered list of sound documents, ranked by relevance to the query. For instance, a user typing "dog" will receive an ordered set of files, where the top ones should contain sounds of barking dogs. Importantly, ordering the sound documents is based solely on acoustic content: no text annotations or other metadata are used at retrieval time. Rather, at training time, a set of annotated sound documents (sound files with textual tags) is used, allowing the system to learn to match the acoustic features of a dog bark to the text tag "dog", and similarly for a large set of potential sound-related text queries. In this way, a small labeled set can be used to enable content-based retrieval from a much larger, unlabeled set.

Several previous studies have addressed the problem of content-based sound retrieval, focusing mostly on the machine-learning and information-retrieval aspects of that task, using standard acoustic representations (Whitman & Rifkin, 2002; Slaney, 2002; Barrington et al., 2007; Turnbull et al., 2008; Chechik et al., 2008). Here we focus on the complementary problem, of finding a good representation of sounds using a given learning algorithm.

The current paper proposes a representation of sounds that is based on models of the mammalian auditory system. Unlike many commonly used representations, it emphasizes fine timing relations rather than spectral analysis. We test this representation in a quantitative task: ranking sounds in response to text queries. This is achieved using a scalable online machine learning approach to ranking. We find that the auditory representation outperforms standard MFCC features, reaching precision above 73% for the top-ranked sound, compared to about 60% for standard MFCC and 67% for the best MFCC variant we found. The following section describes the auditory representation that we use, Section 3 describes the learning approach and Section 4 our experimental results. Our findings are discussed in Section 5.

2 Modeling sounds

In this paper we focus on a class of representations that is partially based on models of the auditory system, and compare these representations to standard mel-frequency cepstral coefficients (MFCCs). The motivation for using auditory models follows from the observation that the auditory system is very effective at identifying many sounds, and this may be partially attributed to the acoustic features that are extracted at the early stages of auditory processing.

We extract features with a four-step process, illustrated in Fig. 1: (1) A nonlinear filterbank with half-wave rectified output. (2) Strobed temporal integration, that yields a *stabilized auditory image* (SAI). (3) Sparse coding using vector quantization. (4) Aggregate all frames features to represent the full audio document.

The first two steps, filterbank and strobed temporal integration, are firmly rooted in auditory physiology and psychoacoustics (Lyon, 1990; Popper & Fay, 1992; Patterson, 2000). The third processing step, sparse coding, is in accordance with some properties of neural coding (Olshausen & Field, 2004), and has significant computational benefits that allow us to train large scale models. The fourth step takes a "bag of features" approach which is common in machine vision and information retrieval. The remainder of this section describes these three steps in detail.

2.1 Cochlear model filterbank

The first processing step is a cascade filterbank inspired by cochlear dynamics, known as the pole–zero filter cascade (PZFC) (Lyon, 1998). It produces a bank of bandpass-filtered, half-wave rectified output signals that simulate the output of the inner hair cells along the length of the cochlea. The PZFC can be viewed as approximating the

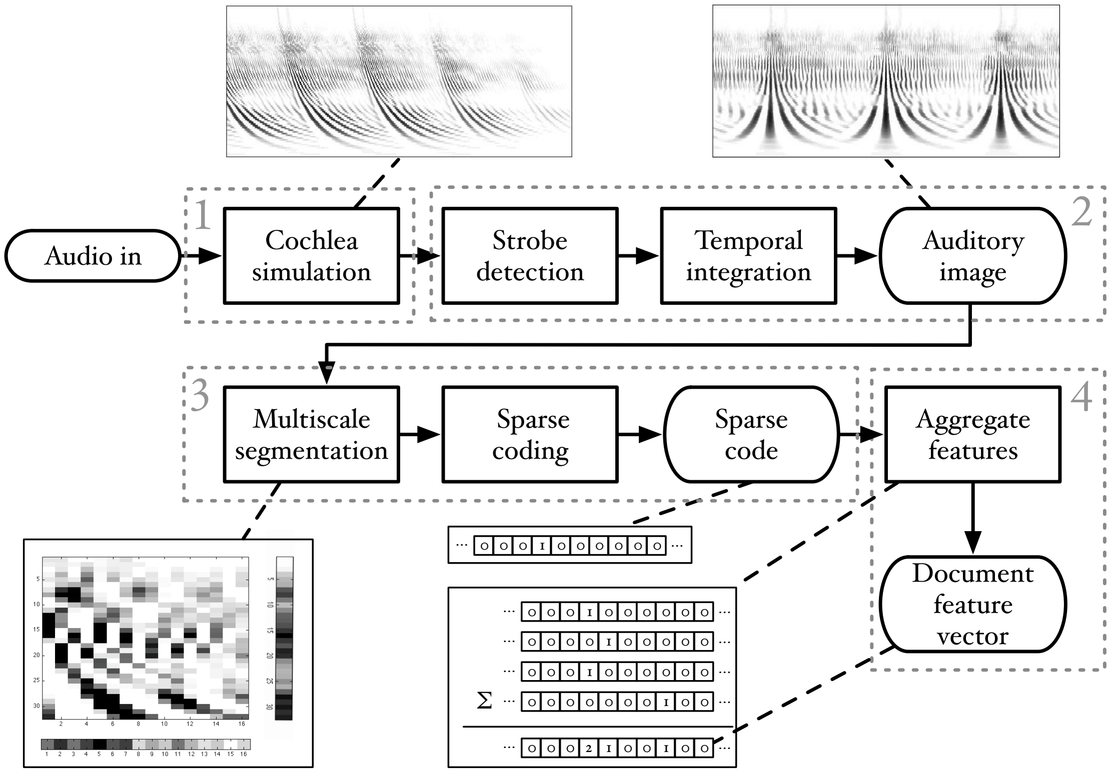

Figure 1: The systems for generating sparse codes from audio using an auditory frontend. It consists of four steps: (1) Cochlea simulation, (2) Stabilized Auditory image creation (3) Sparse coding (4) Aggregate into a document representation.

auditory nerve's instantaneous firing rate as a function of cochlear place, modeling both the frequency filtering and the compressive or *automatic gain control* characteristics of the human cochlea (Lyon, 1990).

More specifically, small segments of the cochlea act as local filters on waves propagating down its length. This local behavior is modeled using a cascade of simple filter stages, each stage defined by just a complex pole pair (a resonance) and a complex zero pair (an anti-resonance). The sound signal is fed into the highest-frequency stage; the output of this stage is passed as the input to the next stage, and so on down the cascade (see Fig. 2). The poles and zeros of each stage are arranged such that the peak gains of the stages go from high frequency to low frequency. The nonlinear mapping of frequency to place is chosen such that a constant increment of place (one filter channel) corresponds to a frequency difference proportional to the psychophysical *equivalent rectangular bandwidth* (ERB) (Glasberg & Moore, 1990). Such nonlinear mappings, including the mel and bark scales, are common in auditory modeling. The pole and zero positioning for the PZFC stages is similar to the "two-pole–two-zero, sharper" arrange-

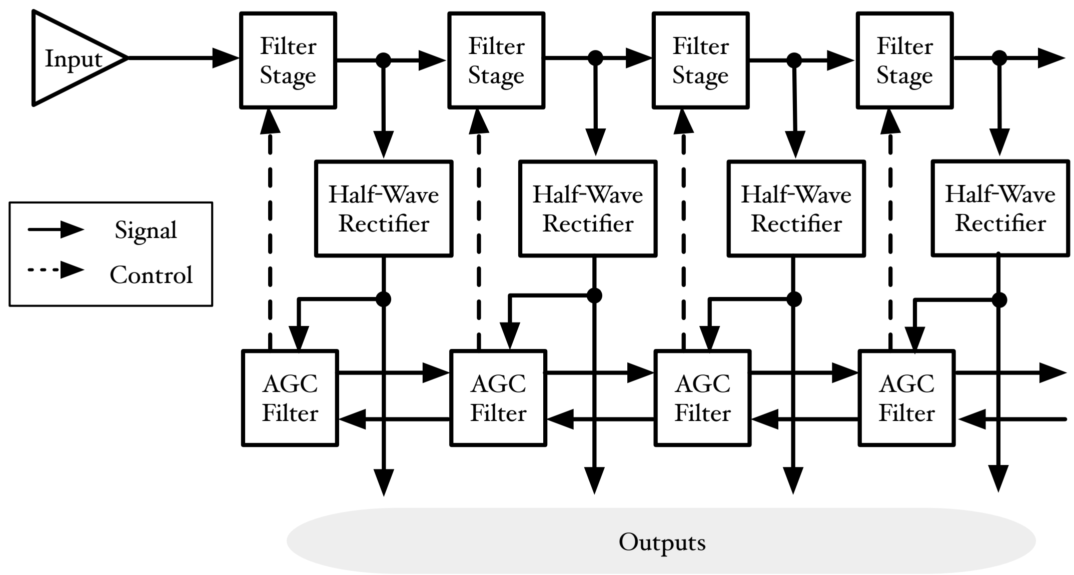

Figure 2: Schematic of the PZFC design. The cascaded filter stages (above) provide a variable gain, which is controlled by the automatic gain control (AGC) smoothing network (below).

ment described in (Lyon, 1998), resulting in a peak gain per stage of only about +5 dB, followed by a valley of about -10 dB. The cascade of many such stages results in a large gain peak followed by a steep cutoff at each tap of the filterbank beyond the first few.

The PZFC also models the adaptive and frequency dependent gain that is observed in the human cochlea, thereby making an *automatic gain control* (AGC) system. Details on this system, including specific parameters of our models are discussed in Appendix A.

2.2 Strobe finding and image stabilization

The second processing step, *strobed temporal integration* (STI), is based on human perception of sounds, rather than purely on the physiology of the auditory system (Patterson & Holdsworth, 1996). In this step, PZFC output is passed through a strobe-finding process, which determines the position of "important" peaks in the output in each channel. These strobe points are used to initiate temporal integration processes in each channel, adding another dimension to represent time delay from the strobe, or trigger, points. Intuitively, this step "stabilizes" the signal, in the same way that the trigger mechanism in an oscilloscope makes a stable picture from an ongoing time-domain waveform.

The end result of this processing is a series of two-dimensional frames of real-valued data (a "movie"), known as a "stabilized auditory image" (SAI). Each frame in this "movie" is indexed by cochlear channel number on the vertical axis and lags relative to identified strobe times on the horizontal axis. Examples of such frames are illustrated in Fig. 3 and Fig. 4.

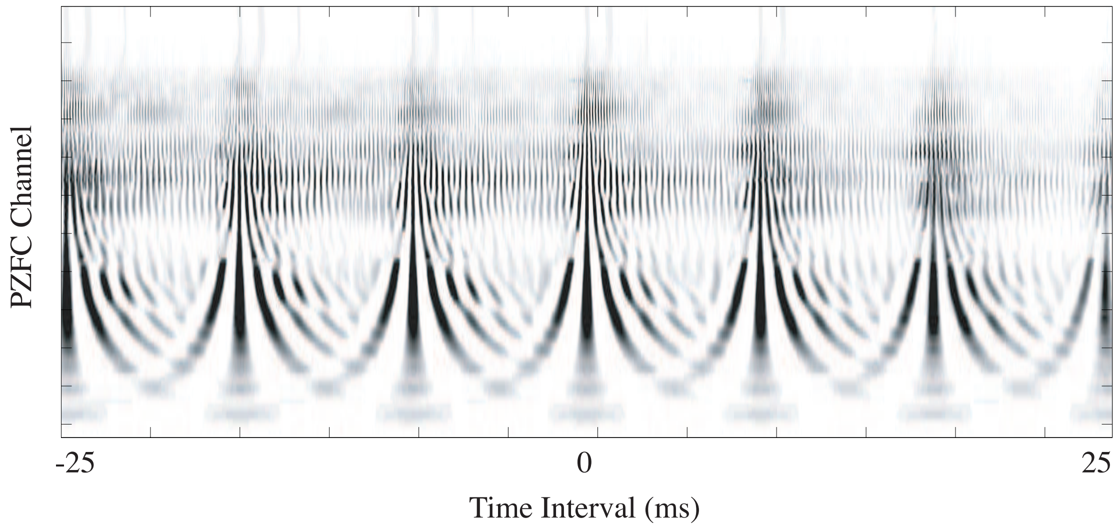

Figure 3: Example of one SAI frame (from an SAI "movie") in response to a human vowel. The moving image will look steady (hence "stabilized") when the audio sounds steady, as is the case with the steady vowel sound.

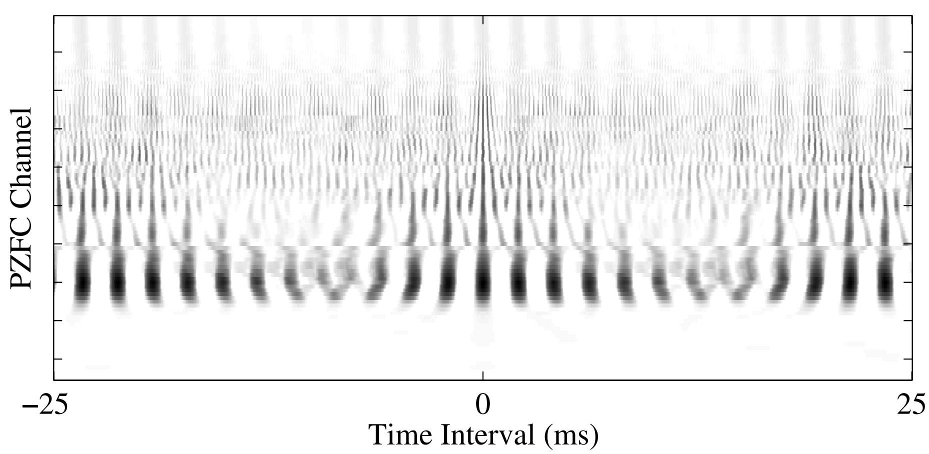

Figure 4: Example of an SAI frame from the sound of a telephone ringing. The picture shows that the sound is less periodic than the voice sound, but has some repeating structure.

The STI process can be viewed as a modified form of autocorrelation. In autocorrelation, a signal is cross-correlated with itself at various delay "lags". The zero-lag is at the center of the output, and the autocorrelation function being symmetrical about the center. In STI, the signal is instead cross-correlated with a sparse function that is zero everywhere except at certain times called strobe points. The height of the signal at these strobe points determines the "weight" with which that time interval is represented in the output. STI is more efficient computationally than autocorrelation, since one of the signals is "sparse". The resulting output is no longer symmetrical about the zero-lag point (Patterson & Irino, 1997). The details of our strobe integration are given in Appendix B.

The filterbank and SAI stages described above represent our current best attempt to combine the good properties of the Slaney/Lyon "correlogram" (Slaney & Lyon, 1993) and the Patterson/Irino "stabilized auditory image." (Patterson & Irino, 1997). The PZFC filterbank can be seen as intermediate between the "passive long-wave" cascade-parallel model and the "active short-wave" all-pole filter cascade of (Slaney & Lyon, 1993), while the stabilization mechanism is closer to Patterson's triggered temporal integration, which maintains time-domain asymmetry in the resulting SAI (Patterson & Irino, 1997; Patterson, 2000), as opposed to the Slaney/Lyon autocorrelogram approach that forces all sounds to produce symmetric images.

2.3 Sparse coding of an SAI

The third processing step transforms the content of SAI frames into a sparse code that captures repeating local patterns in each SA image. Sparse codes have become prevalent in the characterization of neural sensory systems (Olshausen & Field, 2004; Olshausen et al., 1996). A sparse code is a high-dimensional vector $a \in \mathbb{R}^d$ that contains mostly zeros, and only a few non-zero entries $\|a\|_0 = k \ll d$. As such it provides a powerful representation that can capture complex structures in data, while providing computational efficiency. Specifically, sparse codes can focuse on typical patterns that frequently occur in the data, and use their presence to represent the data efficiently.

In a previous work (Chechik et al., 2008) we compared sound ranking systems that use dense and sparse features. The main conclusion from this comparison was that sparse representations obtain a comparable level of retrieval precision, but achieve it with only the fraction of the time needed for training. For instance, training on a dataset of 3431 files took only 3 hours instead of 960 hours (40 days) for training a Gaussian Mixture model. The reason for the improved computational efficiency is (as we show below) that the learning approach we have chosen (PAMIR, see next section) has computational complexity that depends on the number of non-zero values k, rather than the full dimensionality d. Building on these results, this paper focuses on sparse codes only.

A second important aspect of sparse codes, is that they provide a layer of nonlinearity that maps the raw input into a representation that captures typical patterns in the data.

Our sparse code is based on identifying the typical patterns in the set of SAIs, and representing each given SAI frame, or sequence of frames, using a histogram of the patterns that appear in it. This histogram is usually sparse, since each sound typically only contains a relatively small number of patterns. This *bag-of-patterns* representation is similar to the common use of *bag-of-words* representation of text documents, or *bag-of-visual-terms* sometimes used in machine vision. However, unlike machine vision problems in which images are somewhat translation invariant, namely, similar patterns could be found at different parts of an image, the SAI is indexed by frequency and delay lag. As a result, different positions in the SAI correspond to auditory objects that are perceptually different. To handle this, instead of looking for global patterns across the whole SA image, we search for more local patterns at different parts of the SAI. More specifically, the sparse coding step has two sub-steps: First, define a set of overlapping rectangular patches that cover each SAI frame. Second, code each local region using its own sparse-encoder.

For selecting the rectangular local patches, we systematically tried several approaches and tested the precision obtained with each approach in a sound ranking task, as described below. We also tested a few approaches for representing the content of each rectangle, in a compact way. The details are given in Appendix C.

In the second sub-step, we represent all the vectors that represent the rectangular areas in an SAI using sparse codes. We tested two sparse coding approaches: *vector quantization* (VQ) (Gersho & Gray, 1992) and *matching pursuit* (MP) (Bergeaud & Mallat, 1995; Mallat & Zhang, 1993). In VQ, a dense feature vector is approximated by the closest vector from a codebook (in Euclidean sense). Once the best match has been chosen, the representation can be encoded as a sparse code vector, with a length equal to the size of the codebook, that consists of all zeros, except for a single ”one” at the index position of the chosen code word.

In MP, each vector (representing a rectangle) is projected onto the codebook vectors, the largest projection is selected, the signed scalar value of that projection is added to the sparse vector representation (in the appropriate index position), and the vector valued projection is subtracted from the original vector, producing a residual vector. The process is then repeated until the magnitude of the largest projection becomes smaller than a given threshold. For both matching pursuit and vector quantization we learn individual codebooks tailored to represent the rectangles at each specific position in the SAI. The codebook was learned from the full set of rectangles in the data using a standard k-means algorithm, which yields a codebook that is tailored to vector quantization. The problem of finding a codebook that is specifically optimized for MP is very hard, and we chose to use the same codebook for both VQ and MP. To choose the size of the codebook (number of k-means clusters), we tested several values of this parameter. The complete set of codebook sizes tested is described in Appendix D.

Once each rectangle has been converted into a sparse code (using vector quantization or matching pursuit) these codes are concatenated into one very-high-dimensional sparse-code vector, representing the entire SAI frame. With the default parameter set, using vector quantization, a codebook size of 256 was used for each of the 49 rectangles, leading to a feature vector of length $49 \times 256 = 12544$, with 49 nonzero entries.

At each frame time, this feature vector of mostly zeros, with ones (in the VQ case) or amplitude coefficients (in the matching pursuit case) at a sparse set of locations, can be thought of as a histogram of feature occurrences in the frame. To represent an entire sound file, we combine the sparse vectors representing histograms of individual frames into a unified histogram—equivalent to simply adding up all the frame feature vectors. In the interpretation as a histogram, it shows how frequently each abstract feature occurs in the sound file. The resulting histogram vector is still largely sparse and is used to represent the sound file to the learning system described in the following section.

The process described in this section involves multiple parameters. In our experiments, we varied these parameters and tested how they affect the precision of sound ranking. More details are given in Sec. 5 and Sec. D.

3 Ranking sounds given text queries

We now address the problem of ranking sound documents by their relevance to a text query. Practical uses of such a system include searching for sound files or specific moments in the sound track of a movie. For instance, a user may be interested to find vocalizations of monkeys to be included in a presentation about the rain-forest, or to locate the specific scene in a video where a breaking glass can be heard. A similar task is “musical query-by-description”, in which a relation is learned between audio documents and words (Whitman & Rifkin, 2002).

We solve the ranking task in two steps. In the first step, sound documents are represented as sparse vectors, following the procedure described above. In the second step, we train a machine learning system to rank the documents using the extracted features. In a previous study (Chechik et al., 2008), we evaluated different machine learning methods for the second step, while the first step was achieved using standard MFCC features. The methods that we evaluated were Gaussian mixture models (GMM), support vector machines (SVM), and the passive–aggressive model for image retrieval (PAMIR). While all three models achieved similar precision at the ranking task, PAMIR was significantly faster, and the only one that scaled to large data sets. It is therefore suitable for handling large collections of sounds, such as indexing a large fraction of the sound documents on the world wide web. For this reason, in this study we use the PAMIR method as a learning algorithm. The remainder of this section describes the PAMIR learning algorithm (Grangier & Bengio, 2008), recast from the image application to the audio application.

3.1 PAMIR for audio documents

Consider a text query represented by a sparse vector $q \in \mathbb{R}^{d_q}$ where d_q is the number of possible words that can be used in queries (the query dictionary). Also consider a set of audio documents $A \subset \mathbb{R}^{d_a}$, where each audio document is represented as a feature vector, $a \in \mathbb{R}^{d_a}$, and d_a is the dimensionality of the audio feature vector. Let $R(q) \subset A$ be the set of audio documents in A that are relevant to the query q. A ranking system provides a scoring function $S(q, a)$ that allows ranking of all documents $a \in A$ for any given query q. An ideal scoring function would rank all the documents $a \in A$ that are relevant for q ahead of the irrelevant ones:

$$S(q, a^+) > S(q, a^-) \quad \forall a^+ \in R(q), a^- \in \overline{R}(q) \ . \tag{1}$$

where $\overline{R}(q)$ is the set of sounds that is not relevant to q. The simplest score of PAMIR uses a bilinear parametric score:

$$S_{\mathbf{W}}(q, a) = q^T \mathbf{W} a \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{d_q \times d_a}$. The matrix $\mathbf{W}$ can be viewed as a linear mapping from audio features to query words. Namely, the product $\mathbf{W}a$ is viewed as a "bag of words" description of the audio document, and the dot product of this bag of words with the query words q gives the score.

When q and a are sparse, the score $S_{\mathbf{W}}$ can be computed very efficiently even when the dimensions d_a and d_q are large. This is because the matrix multiplication only requires $O(|q||a|)$ operations where $|q|$ and $|a|$ are the number of non-zero values in q and a respectively.

To learn the scoring function $S_{\mathbf{W}}$ we use an algorithm based on the passive–aggressive (PA) family of learning algorithms introduced by (Crammer et al., 2006). Here we consider a variant that uses triplets (q_i, a_i^+, a_i^-), consisting of a text query and two audio documents: one that is relevant to the query, $a_i^+ \in R(q_i)$, and one that is not, $a_i^- \in \bar{R}(q_i)$.

The learning goal is to tune the parameters $\mathbf{W}$ of the scoring function such that the relevant document achieves a score that is larger than the irrelevant one, with a safety margin:

$$S_{\mathbf{W}}(q_i, a_i^+) > S_{\mathbf{W}}(q_i, a_i^-) + 1 \quad \forall \left(q_i, a_i^+, a_i^-\right) \ . \tag{3}$$

To achieve this goal we define the hinge loss function for all triplets:

$$\begin{aligned} L_{\mathbf{W}} &= \sum_{(q_i, a_i^+, a_i^-)} l_{\mathbf{W}}(q_i, a_i^+, a_i^-) && (4) \\ l_{\mathbf{W}}(q_i, a_i^+, a_i^-) &= \max\left(0, 1 - S_{\mathbf{W}}(q_i, a_i^+) + S_{\mathbf{W}}(q_i, a_i^-)\right) . \end{aligned}$$

The sum in $L_{\mathbf{W}}$ is typically over a set that is too large to be evaluated, but we can use an online algorithm that nevertheless converges to its minimum. We first initialize $\mathbf{W}_0$ to 0. Then, the algorithm follows a set of optimization iterations. At each training iteration i, we randomly select a triplet (q_i, a_i^+, a_i^-), and solve the following convex optimization problem with soft margin:

$$\mathbf{W}_i = \operatorname*{argmin}_{\mathbf{W}} \ \frac{1}{2}\|\mathbf{W} - \mathbf{W}_{i-1}\|^2_{Fro} + C l_{\mathbf{W}}(q_i, a_i^+, a_i^-) \tag{5}$$

where $\|\cdot\|_{Fro}$ is the Frobenius norm. At each iteration i, optimizing $\mathbf{W}_i$ achieves a trade-off between remaining close to the previous parameters $\mathbf{W}_{i-1}$ and minimizing the loss on the current triplet $l_{\mathbf{W}}(q_i, a_i^+, a_i^-)$. The *aggressiveness* parameter C controls this trade-off. The problem in Eq. (5) can be solved analytically and yields a very efficient parameter update rule, described in Fig. 5. The derivation of this analytical solution follows closely the derivation of the original passive–aggressive algorithm (so named because the update rule is passive when the hinge loss term is already zero, and aggressively tries to make it zero when it is not).

In practice, the hyper parameter C can be set using cross validation. For a stopping criterion, it is a common practice to continuously trace the performance of the trained model on a held out set, and stop training when this performance no longer improves.

PAMIR to Rank Audio Documents from Text Queries

Initialization:

Initialize $\mathbf{W}_0 = 0$

Iterations

repeat

Sample a query q_i, and audio documents a_i^+ and a_i^-,

such that $a^+ \in R(q_i) > a^- \in \bar{R}(q_i)$.

Update $\mathbf{W}_i = \mathbf{W}_{i-1} + \tau_i \mathbf{V}_i$

where $\tau_i = \min \left\{ C, \frac{l_{\mathbf{W}_{i-1}}(q_i, a_i^+, a_i^-)}{\|\mathbf{V}_i\|^2} \right\}$

and $\mathbf{V}_i = q_i (a_i^+ - a_i^-)^T$

until (stopping criterion)

Figure 5: Pseudo-code of the PAMIR algorithm. The subscript i is the iteration index. The matrix V_i is the outer product $V_i = ([q_i^1(a_i^+ - a_i^-), \ldots, q_i^{d_q}(a_i^+ - a_i^-)]^T$, where the superscripts on q_i indicate selected components of the query vector.

Sampling a triplet can be done efficiently: We keep a list of audio documents that are relevant for each text query. Given a text query, this allows us to sample uniformly among all the relevant documents. To sample an irrelevant audio document a^-, we repeatedly sample an audio document from the set of all audio documents until we find one that is not relevant to the given query. Since our data has significantly more irrelevant documents than relevant document for any query, an irrelevant audio document can be found with high probability within a few iterations (typically one).

4 Experiments

We evaluate the auditory representation in a quantitative ranking task using a large set of audio recordings that cover a wide variety of sounds. We compare sound retrieval based on the SAI with standard MFCC features. In what follows we describe the dataset and the experimental setup.

4.1 The dataset

We collected a data set that consists of 8638 sound effects from multiple sources. Close to half of the sounds (3855) were collected from commercially available sound effect collections. Of those, 1455 are from the BBC sound effects library. The remaining 4783 sounds are taken from a variety of web sites; *www.findsounds.com*, *partners in rhyme*, *acoustica.com*, *ilovewavs.com*, *simplythebest.net*, *wav-sounds.com*, *wavsource.com*, and *wavlist.com*. Most of the sounds contain only a single "auditory object", and contain the "prototypical" sample of an auditory category. Most sounds are a few seconds long but there are a few that extend to several minutes.

We manually labeled all of the sound effects by listening to them and typing in a handful of tags for each sound. This was used for adding tags to existing tags (from *www.findsounds.com*) and to tag the non-labeled files from other sources. When labeling, the original file name was displayed, so the labeling decision was influenced by the description given by the original author of the sound effect. We restricted our tags to a somewhat limited set of terms. We also added high level tags to each file. For instance, files with tags such as 'rain', 'thunder' and 'wind' were also given the tags 'ambient' and 'nature'. Files tagged 'cat', 'dog', and 'monkey' were augmented with tags of 'mammal' and 'animal'. These higher level terms assist in retrieval by inducing structure over the label space. All terms are stemmed, using the Porter stemmer for English. After stemming, we are left with 3268 unique tags. The sound documents have an average of 3.2 tags each.

4.2 The Experimental Setup

We used standard cross validation to estimate performance of the learned ranker. Specifically, we split the set of audio documents in three equal parts, using two thirds for training and the remaining third for testing. Training and testing was repeated for all three splits of the data, such that we obtained an estimate of the performance on all the documents. We removed from the training and the test set queries that had fewer than $k = 5$ documents in either the training set or the test set, and removed the corresponding documents if these contained no other tag.

We used a second level of cross validation to determine the values of the hyper parameters: the aggressiveness parameter C, and the number of training iterations. In general performance was good as long as C was not too high, and lower C values required longer training. We selected a value of $C = 0.1$, which was also found to work well in other applications (Grangier & Bengio, 2008), and 10M iterations. From our experience the system is not very sensitive to the value of these parameters.

To evaluate the quality of the ranking obtained by the learned model we used the precision (fraction of positives) within the top k audio documents from the test set as ranked for each query.

4.3 SAI and sparse coding parameters

The process of transformation of SAI frames into sparse codes has several parameters which can be varied. We defined a default parameter set and then performed experiments in which one or a few parameters were varied from this default set.

The default parameters cut the SAI into rectangles starting with the smallest size of 16 lags by 32 channels, leading to a total of 49 rectangles. All the rectangles were reduced to 48 marginal values each, and for each box a codebook of size 256, for a total of $49 \times 256 = 12544$ feature dimensions, as described in Sec. 2.3.

Using this default experiment as a baseline for comparisons, we made systematic variations to several parameters and studied their effect on the retrieval precision. First, we modified two parameters that determine the shape of the PZFC filter: *Pdamp* and *Zdamp*. Then, we modified the smallest rectangle size used for sparse segmentation and by limiting the maximum number of rectangles used for the sparse segmentation (with variations favoring smaller rectangles and larger rectangles). Further variants used systematic variation of the codebook sizes used in sparse coding (using both standard vector quantization and matching pursuit). The values of all the experimental parameters used are tabulated in Appendix D.

4.4 Comparisons with MFCC

We used standard Mel frequency cepstral coefficients (MFCC), which we turned into a sparse code in the same way as for the SAIs. MFCCs were computed using a Hamming window. We added the first and second derivatives as additional features of each frame ("delta" and "delta-delta"). We set the MFCC parameters based on a configuration that was optimized for speech, and further systematically varied three parameters of the MFCCs: the number of cepstral coefficients (traditionally 13 for speech), the length of each frame (traditionally 25ms) and the number of codebooks used to sparsify the MFCC of each frame. Optimal performance was obtained with a single codebook of size 5000, 40ms frames and 40 cepstral coefficients (see Sec. 5). Using a single codebook was different from the multi-codebook approach that we took with the SAIs. The optimal configuration corresponds to much higher frequency resolution than the standard MFCC features used for speech.

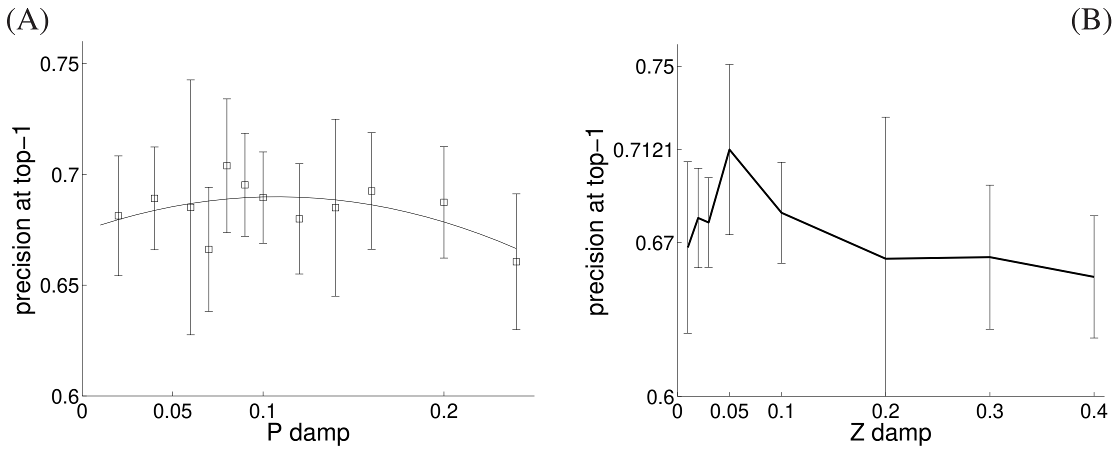

Figure 6: **(A)** Precision at top 1 as a function of the *Pdamp* parameter. **(B)** Precision at top 1 as a function of the *Zdamp* parameter.

5 Results

We first performed a series of experiments to identify the set of parameters that achieve the most accurate sound retrieval. We started by testing the effect of several PZFC parameters on test-set retrieval precision. Specifically, we looked at te effect of two parameters that determine the shape of the PZFC filter, *Pdamp* and *Zdamp*, whose effect is discussed in details in section A. Figure 6(A) plots precision at top-1 retrieved result, averaged over all queries, as a function of the *Pdamp* parameter, with *Zdamp* fixed at the baseline value of 0.2 (see Fig. 11 to understand how these parameters affect the shape of the filter). Precision is quite insensitive to the value of the parameter, with an optimum obtained near 0.12, our baseline value. This parameter sets the pole damping in the small-signal limit (that is, for very quiet sound input), and the AGC largely takes out the effect of this parameter, so the relative insensitivity is not surprising, and corresponds roughly to a relative insensitivity to input sound level or gain.

We also tested the effect of the *Zdamp* parameter, the zero damping, which remains fixed, independent of the AGC action. Precision is best for a value near 0.06, which is less than the baseline value of 0.2 used in other tests, but the difference is not highly significant. The lower *Zdamp* corresponds to a steeper high-side cutoff of the filter shapes. Figure 6(B) plots precision as a function of the *Zdamp* parameter.

We further tested the effect of various parameters of the SAI feature extraction procedure on the test-set precision. Fig. 7 plots the precision of the top ranked sound file against the length of the sparse feature vector, for all our experiments. Each set of experiments has their own marker. For instance, the series of stars show precision for a set

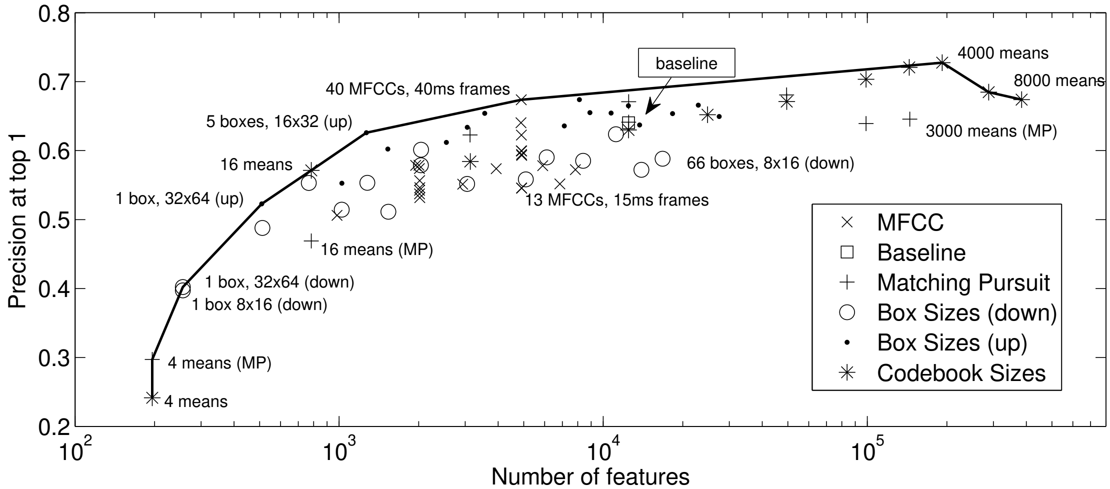

Figure 7: Ranking precision at the top-1 sound plotted against feature count, for all experiments run. Selected experiment names are plotted on the figure near each point. The different experiment sets are denoted by different markers. The convex hull joining the best-performing points is plotted as a solid line. *means* stands for the size of the dictionary (number of centroids used in the k-means algorithm). *MP* stands for Matching Pursuit.

of experiments where the number of means (size of the codebook) is varied. The rest of the parameters do not change from one star to the other, and were set at the default parameters defined in Sec. 2.3.

top-k	SAI	MFCC	percent error reduction
1	27	33	18%
2	39	44	12%
5	60	62	4%
10	72	74	3%
20	81	84	4%

Table 1: Comparison of error at top-k for best SAI and MFCC configurations (error defined as one minus precision).

Interestingly, performance saturates with a very large number of features $\sim 10^5$, resulting from using 4000 code words per codebook, and a total of 49 codebooks. This parameter configuration achieved 73% at the top ranked sound file, which was significantly better than the best MFCC result which achieved 67% (wilcoxon test for equal medians p-value = 0.0078). This reflects about 18% smaller error (from 33% to 27% error). SAI features also achieve better precision-at-top-k consistently for all values of

k, although with lower relative precision improvement (Tab. 1). It should be stressed however that the parameters that we found (and the auditory model architecture in general) are not guaranteed to be "optimal", and it is possible that further refinement could further improve the retrieval precision. Also, the relatiely poor performance of MP could be due to the fact that the dictiontionary we used may not be optimal for MP.

Importantly, although the best sparse codes use very high dimensional vectors, the actual data lives in a much lower dimensional subspaces. Specificaly, eventhough each SAI is represented by a vector of length 49×4000, only 49 values in this representation are non-zero. Since subsequent frames have similar characteristics, the overall representation of a sound file typically has only a few hundreds of non zero values.

Table 2 shows three sample queries together with the top 5 test sound documents returned by the best SAI-based and MFCC-based trained systems. Documents that were labeled as relevant are marked with [R]. The three queries shown were selected such that both systems performed significantly differently on them, as shown in Fig. 8. While being a very small sample of the data, it shows that both systems behave reasonably well, most often returning good documents, or at least documents that appear not-too-far from the expected answer. For instance, the SAI-based system returns document "water-dripping" for the query "gulp", which, while being wrong, is admittedly not far from the mark. Similarly, the document "45-Crowd-Applause" is returned by the MFCC-based system for query "applaus-audienc", despite not being labeled as relevant for that query.

The performance that we calculated was based on textual tags, which are often noisy and incomplete. In particular, people may use different terms to describe very similar concepts. Also, the same sound may be described across different aspects. For instance a music piece may be described by the playing instrument ("piano") or the mood it conveys ("soothing") or the name of the piece. This multi-label problem is common in content based retrieval, being shared by image search engines, for example.

Table 3 shows queries that consistently "confused" our system and caused retrieval of sounds with a different label. For each pair of queries q_1 and q_2 we measure confusion, by counting the number of sound files that were ranked within the top-k files for query q_1, but not for q_2 even though q_2 was identical to their labels. For example, there were 7 sound files that were labeled *evil laugh* but were not ranked within the top k documents for the query *evil laugh*, and at the same time ranked highly for *laugh*.

As can be seen from the table, the repeated "mistakes" are typically due to labeling imprecision: when a sound labeled *laugh* is retrieved for a query *evil laugh*, the system

Query	SAI file (labels)	MFCC file (labels)
tarzan	Tarzan-2 (tarzan, yell) [R]	TARZAN (tarzan, yell) [R]
	tarzan2 (tarzan, yell) [R]	175orgs (steam, whistl)
	203 (tarzan) [R]	mosquito-2 (mosquito)
	wolf (mammal, wolves, wolf, ...)	evil-witch-laugh (witch, laugh, evil)
	morse (mors, code)	Man-Screams (horror, scream, man)
applaus	27-Applause-from-audience [R]	26-Applause-from-audience [R]
audienc	30-Applause-from-audience [R]	phaser1 (trek, phaser, star)
	golf50 (golf)	fanfare2 (fanfar, trumpet)
	firecracker	45-Crowd-Applause (crowd, applaus)
	53-ApplauseLargeAudienceSFX [R]	golf50
gulp	tite-flamn (hit, drum, roll)	GULPS (gulp, drink) [R]
	water-dripping (water, drip)	drink (gulp, drink) [R]
	Monster-growling	california-myotis-search (blip)
	(horror, monster, growl)	
	Pouring (pour, soda)	jaguar-1 (bigcat, jaguar, mammal,...)

Table 2: Top documents obtained for queries that performed very differently between the SAI and MFCC feature based systems.

counts it as a mistake, even though this is likely to be a relevant match. In general we find that confused queries are often semantically similar to the sound label, hence the errors made by the ranking systems actually reflect the fact that the sound files have partial or inconsistent labeling. This demonstrates a strength of content-based sound ranking: it can identify relevant sounds even if their textual labels are incomplete, wrong or maybe even maliciously spammed.

query, label		total number of errors (SAI + MFCC)
clock-tick	cuckoo	8
door knock	door	8
evil laugh	laugh	7
laugh witch	laugh	7
bell-bicycl	bell	7
bee-insect	insect	7

Table 3: Error analysis. Queries that were repeatedly confused for another query. All pairs of true-label and confused labels with total count above seven are listed.

Fig. 8 summarizes the performance of the SAI and MFCC systems with respect to their average precision on the set of all test queries. The graph shows queries either by their label or a “plus” sign (showing all queries would have resulted in a very difficult figure to read), positioned according to the performance of each system with respect to that query, over the test set. As can be seen, there are more queries above the separating line than below, which means that overall the SAI-based system yielded a better average precision than the MFCC-based system.

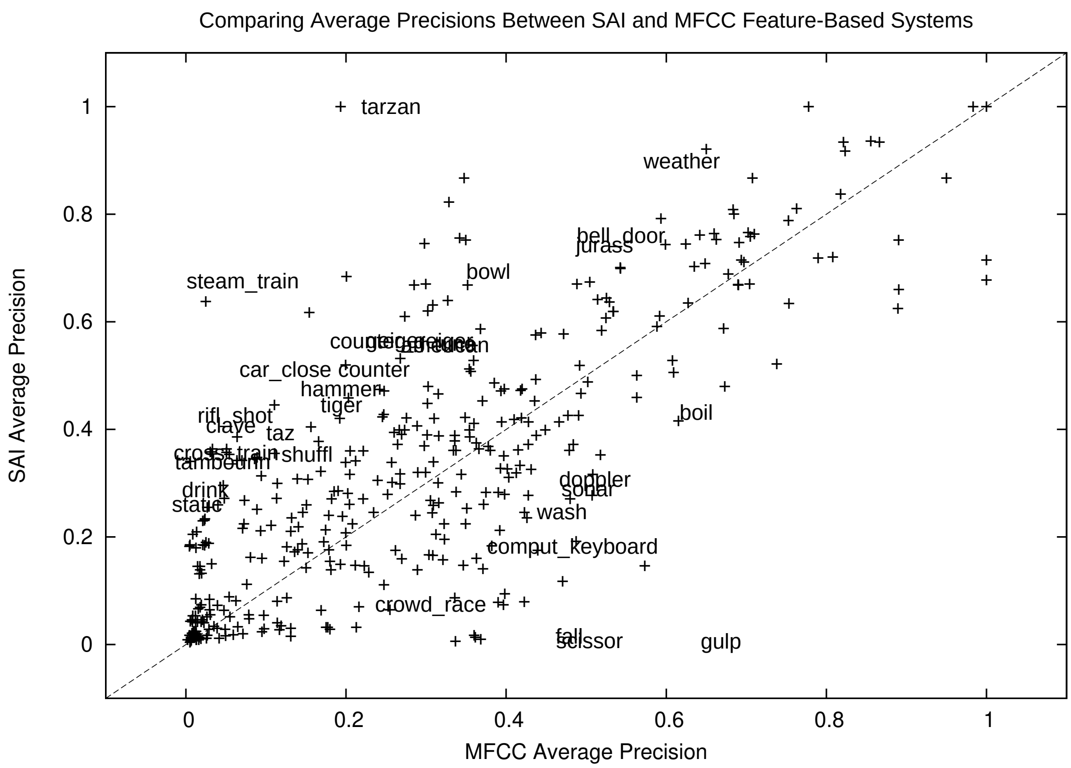

Figure 8: Comparison of average precisions between SAI and MFCC based systems. Each ‘plus’ sign or label corresponds to a single query, which coordinates provides the test average precisions for both systems. Queries are stemmed and spaces have been replaced by underscores. More queries appear above the $y = x$ line, showing that the SAI based system obtained a high mean average precision.

6 Conclusion

We described a content-based sound ranking system that uses biologically inspired auditory features and successfully learns a matching between acoustics and known text labels.

We have previously shown that the PAMIR learning algorithm, applied over a sparse representation of sounds (Chechik et al., 2008) is an efficient framework for large-scale retrieval and ranking of media documents from text queries. Here we used PAMIR to study systematically many alternative sparse-feature representations ("front ends").

Our findings support the hypothesis a front end that mimics several aspects of the human auditory system provides an effective representation for machine hearing. These aspects include a realistic nonlinear adaptive filterbank and a stage that exploits temporal fine structure at the filterbank output (modeling the cochlear nerve) via the concept of the stabilized auditory image. Importantly however, the auditory model described in this paper may not be always optimal, and future work on characterizing the optimal parameters and architecture of auditory models is expected to further improve the precision, depending on the task at hand.

One approach to feature construction would have been to manually construct features that are expected to discriminate well between specific classes of sounds. For instance, periodicity could be a good discriminator between wind in the trees and a howling wolf. However, as number of classes grows, such careful design of discriminative features may become infeasible. Here we take an opposite approach, assuming that perceptual differences rely on lower level cochlear feature extraction, we use models inspired by cochlear processing to obtain a very high dimensional representation, and let the learning algorithm identify the features that are most discriminative.

The auditory model representation we use do not take into account long-term temporal relationships, by analogy with the "bag of words" approach common in text document retrieval. The SAI features capture a short window of time, up to about 50 msec for the biggest box features, which is comparable to but a bit less than that captured by MFCC with double deltas. Longer correlations are likely to be also useful for sound retrieval and it remains an interesting research direction to study how they should be estimated and used.

The auditory models that we used to represent sounds involve multiple nonlinear complex components. This could be one reason why the features generated using this system are more discriminative than standard MFCC features for a test database of sounds. However, it is hard to assess what aspects of our test sounds are better represented by the features from the auditory model.

One difference between SAI and MFCC representations is that SAIs retain fine timing information, while MFCC preserves fine spectral structure, at least when the number of coefficients is large. However, the two representations differ by numerous other properties, including the cepstral transform, and the AGC. Further study will be needed to gain more specific insights into exactly what properties of the different auditory models and sparse feature extraction schemes are the key to good performance.

Since our system currently uses only features from short windows, we envision future work to incorporate more dynamics of the sound over longer times, either as a *bag-of-patterns* using patterns that represent more temporal context, or through other methods. We also envision future work to test our hypothesis that, in our retrieval task, the system using local SAI patterns will be more robust to interfering sounds than a system using more traditional spectral features such as MFCC.

References

Barrington, L., Chan, A., Turnbull, D., & Lanckriet, G. (2007). Audio information retrieval using semantic similarity. *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*. IEEE.

Bergeaud, F., & Mallat, S. (1995). Matching pursuit of images. *Image Processing, 1995. Proceedings., International Conference on*.

Bottou, L., Chapelle, O., Decoste, D., & Weston, J. (2007). *Large-scale kernel machines*. MIT Press.

Chechik, G., Ie, E., Rehn, M., Bengio, S., & Lyon, D. (2008). Large-scale content-based audio retrieval from text queries. *ACM International Conference on Multimedia Information Retrieval, MIR*. Vancouver, Canada.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*, *7*, 551–585.

Gersho, A., & Gray, R. (1992). *Vector quantization and signal compression*. Springer.

Glasberg, B. R., & Moore, B. C. J. (1990). Derivation of auditory filter shapes from notched noise data. *Hearing Research*, *47*, 103–138.

Grangier, D., & Bengio, S. (2008). A discriminative kernel-based model to rank images from text queries. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, *30*, 1371–1384.

Irino, T., & Patterson, R. (2006). A dynamic compressive gammachirp auditory filterbank. *Audio, Speech, and Language Processing, IEEE Transactions on*, *14*, 2222–2232.

Krumbholz, K., Patterson, R. D., & Pressnitzer, D. (2000). The lower limit of pitch as determined by rate discrimination. *The Journal of the Acoustical Society of America*, *108*, 1170–1180.

Lopez-Poveda, E. A., & Meddis, R. (2001). A human nonlinear cochlear filterbank. *The Journal of the Acoustical Society of America*, *110*, 3107–3118.

Lyon, R. F. (1982). A computational model of filtering, detection, and compression in the cochlea. *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)* (pp. 1282–1285). Paris, France: IEEE.

Lyon, R. F. (1990). Automatic gain control in cochlear mechanics. In D. P. Geisler, C. D. M. J. W., R. M. A. and S. C. R. (Eds.), *The mechanics and biophysics of hearing*, 395–402. Springer-Verlag.

Lyon, R. F. (1998). Filter cascades as analogs of the cochlea. In T. S. Lande (Ed.), *Neuromorphic systems engineering: neural networks in silicon*, 3–18. Norwell, MA, USA: Kluwer Academic Publishers.

Lyon, R. F., & Mead, C. (1988). An analog electronic cochlea. *IEEE Transactions on Acoustics Speech and Signal Processing*, *36*, 1119–1134.

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

Olshausen, B. A., et al. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*, 607–609.

Olshausen, B. A., & Field, D. J. (2004). Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, *14*, 481–487.

Patterson, R. D. (2000). Auditory images: How complex sounds are represented in the auditory system. *The Journal of the Acoustical Society of America*, *21*, 183–190.

Patterson, R. D., & Holdsworth, J. (1996). A functional model of neural activity patterns and auditory images. *Advances in Speech, Hearing and Language Processing*, *3*, 547–563.

Patterson, R. D., & Irino, T. (1997). Auditory temporal asymmetry and autocorrelation. *Proceedings of the 11th International Symposium on Hearing: Psychophysical and physiological advances in hearing*. Whurr, London.

Popper, A. N., & Fay, R. R. (1992). *The mammalian auditory pathway: Neurophysiology*. Springer.

Slaney, M. (2002). Semantic-audio retrieval. *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)* (pp. 4108–4111). Orlando Florida: IEEE.

Slaney, M., & Lyon, R. (1993). On the importance of time—a temporal representation of sound. In M. Cooke, S. Beet and M. Crawford (Eds.), *Visual representations of speech signals*, 95–116. John Wiley and Sons.

Turnbull, D., Barrington, L., Torres, D., & Lanckriet, G. (2008). Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio Speech and Language Processing*, *16*, 467.

Unoki, M., Irino, T., Glasberg, B., Moore, B., & Patterson, R. (2006). Comparison of the roex and gammachirp filters as representations of the auditory filter. *The Journal of the Acoustical Society of America*, *120*, 1474–1492.

Whitman, B., & Rifkin, R. (2002). Musical query-by-description as a multiclass learning problem. *IEEE Workshop on Multimedia Signal Processing* (pp. 153–156).

A Automatic gain control in PZFC

The damping of the pole pair in each stage is varied to modify the peak gain (and to a lesser extent the bandwidth) of that stage, down to about +1 dB. This system of gain and bandwidth modification by manipulation of pole dampings (or Q values) is a refinement of that described in (Lyon & Mead, 1988; Slaney & Lyon, 1993), which used an all-pole filter cascade, and in (Lyon, 1998), which introduced the two-pole–two-zero stage, which we now call the PZFC. The poles are modified dynamically by feedback from a spatial/temporal loop filter, or smoothing network, thereby making an *automatic gain control* (AGC) system. The smoothing network takes the half-wave rectified output of all channels, applies smoothing both in the time and cochlear place dimensions, and uses both global and local averages of the filterbank response to proportionately increase the pole damping of each stage. This coupled AGC smoothing network descends from one first described in (Lyon, 1982) (in that work, the loop filter directly controlled a post-filterbank gain rather than a pole damping as in the present work). The PZFC filterbank architecture can be seen as intermediate between the all-pole filter cascade (Slaney & Lyon, 1993) on one hand and cascade-parallel models on the other hand. The compression exhibited by the PZFC includes both a fast-acting AGC part, similar to that of the "dynamic compressive gammachirp" (Irino & Patterson, 2006) and an instantaneous part, from an odd-order nonlinearity similar to that in the "dual-resonance, nonlinear" (DRNL) model (Lopez-Poveda & Meddis, 2001).

The complex transfer function of one stage of the filter cascade is a rational function of the Laplace transform variable s, of second order in both numerator and denominator, corresponding to a pair of zeros (roots of the numerator) and a pair of poles (roots of the denominator):

$$H(s) = \frac{s^2/\omega_z^2 + 2\zeta_z s/\omega_z + 1}{s^2/\omega_p^2 + 2\zeta_p s/\omega_p + 1} \tag{6}$$

where ω_p and ω_z are the natural frequencies, and ζ_p and ζ_z are the damping ratios, of the poles and zeros, respectively. The natural frequencies are constants, decreasing from each stage to the next, along a cochlear frequency–place map, or ERB-rate scale (Glasberg & Moore, 1990); for the present experiments, at each stage ω_z is fixed at $1.4\omega_p$, and the ω_p decreases by one-third of the ERB at each stage. The filters are implemented as discrete-time approximations at 22050 Hz sample rate (f_s) by mapping the poles and zeros to the z plane using $z = e^{s/f_s}$ as is conventional in the simple "pole–zero mapping" or "pole–zero matching" method of digital filter design.

Two parameters of the PZFC, *Pdamp* and *Zdamp*, are of special interest since they determine the shape of the filter's transfer function by setting the damping ratios ζ_p and ζ_z. The *Zdamp* parameter directly sets the zero damping

$$\zeta_z = Zdamp \tag{7}$$

while the pole damping varies dynamically as

$$\zeta_p = (1 + AGC)Pdamp \tag{8}$$

where AGC is an automatic-gain-control feedback term proportional to the smoothed half-wave rectified filterbank output. In other versions of the model (not tried in the present work) ζ_z may also vary with the AGC feedback.

Fig. 11(A) shows the effect of the varying AGC term on the filter gain magnitude of a single filter stage, in the baseline case of $Pdamp = 0.12$ and $Zdamp = 0.2$.

In the complex s plane of Fig. 11(B), the damping ratio ζ is defined as the distance of the poles or zeros from the imaginary s axis, relative to their distance ω from the origin. The ζ_p, and therefore the *Pdamp* parameter, affects mainly the width and peak gain of the filter, with lower damping values giving narrower and higher-gain filters. The *Zdamp* parameter sets the damping of the zeros and affects the symmetry of the filter, with lower values giving steeper high-side cutoff.

Fig. 11(C) shows the effect of these two parameters on the shape of the filter cascade's gain magnitude. The apparent large variation in peak gain is partially compensated by the AGC action that adjusts pole damping and compresses the output level variation.

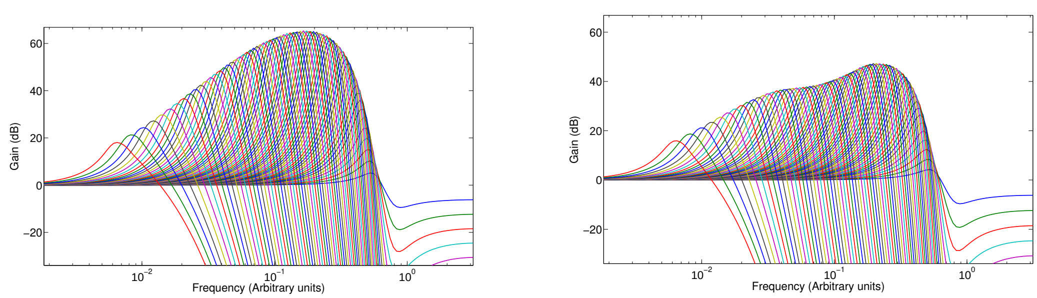

Figure 9: Adaptation of the overall filterbank response at each output tap. The plot on the left shows the initial response of the filterbank before adaptation. The plot on the right shows the response after adaptation to a human /a/ vowel of 0.6 sec duration. The plots show that the adaptation affects the peak gains (the upper envelope of the filter curves shown), while the tails, behaving linearly, remain fixed.

Fig. 9 show the transfer function gain of the all the outputs of the filter cascade, in the case of silence and as adapted to a vowel sound at moderate level.

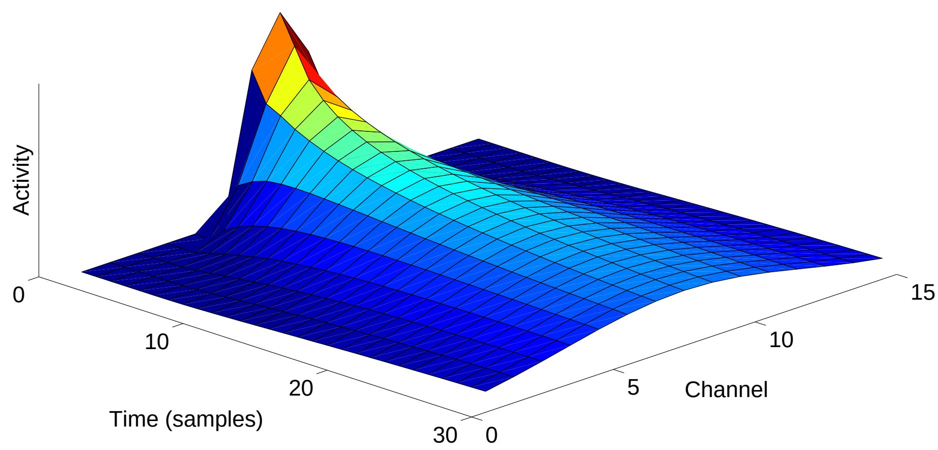

Figure 10: Impulse response of the smoothing network used as the AGC loop filter of the PZFC. An impulse in one channel spreads out into adjacent channels and dies away over time. There are four such filters in parallel in the AGC, each with a different decay constant; the slower ones have more spatial spread before the signal decays. The filter adaptation, controlled by pole damping, in a channel of the PZFC is based on a weighted sum of the activities from each of the four smoothing filters at that channel.

Fig. 10 shows the impulse response in time and space of the smoothing filter in the automatic-gain-control loop that averages the half-wave rectified outputs of the filterbank to produce the AGC signal that controls the pole damping in each channel. The multiple channels of these feedback filters are coupled through the space dimension (cochlear place axis), such that each channel's damping is affected by the output of other nearby channels.

All the filterbank parameters were chosen to obtain a filterbank behavior that is roughly similar to that of the human cochlea, but parameters were not directly learned from data. The filter bandwidths and shapes are close to those of the asymmetric "gammachirp" filters that have been fitted to human data on detection of tones in notched-noise maskers (Unoki et al., 2006). The detailed values of all parameters are provided with the accompanying code that we make available online at *http://www.acousticscale.org/link/AIM-PZFC*.

B Strobed temporal Integration

To choose strobe points, the signal in each channel is multiplied point-wise by a windowing function, a parabola of 40ms width. The maximum point in the windowed

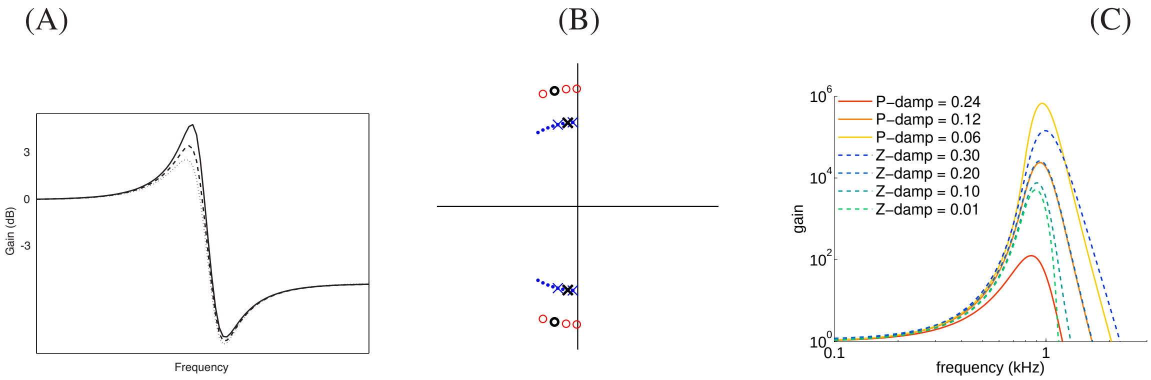

Figure 11: **(A)** Frequency response of a single PZFC stage as the pole damping and stage peak gain change in response to sound level via the AGC feedback changing the pole damping; the peak gain changes by only a few dB. **(B)** The pole and zero positioning for a single stage of the PZFC, in the complex s plane; crosses show starting positions of the poles at low signal levels; circles show the fixed positions of the zeros. The poles move dynamically along the dotted paths (on circles of constant natural frequency) as their Q or damping parameters are changed, thereby varying the filter peak gain. The low-frequency tail gain remains fixed, passing lower frequencies transparently to later stages. The pole and zero positions shown by heavy black symbols represent a default model that we study in the experiments below as a "baseline model". The lighter color circles represent alternatives to that default, obtained by varying the *Pdamp* and *Zdamp* parameters. The ratio between the natural frequency (distance from the origin) of the zeros and that of the poles is fixed at 1.4, putting the dip in the response about a half octave above the peak frequency. **(C)** The magnitude transfer function, in the small-signal limit, of one channel (the output after a cascade of 60 stages in this example) of the PZFC cochlear filterbank, with center frequency near 1 kHz, for several values of *Pdamp* and *Zdamp*. The solid red–orange curves show variation of *Pdamp* through values 0.06 (highest curve), 0.12, and 0.24. The dashed blue–green curves show variation of *Zdamp*, for values 0.01 (lowest dashed curve), 0.1, 0.2 and 0.3. The dashed and solid curves coincide for the baseline parameters.

signal is the strobe point. The window is then shifted by 4ms and the process is repeated. Thus, there is guaranteed to be an average of one strobe point every 4ms, or five per 20ms frame; it is possible for multiple strobes to occur at one point in the signal, since the windows overlap. Each SAI channel is then calculated as the cross-correlation between the original signal and a signal composed of delta-functions at the identified strobe points. This cross-correlation is accomplished efficiently by simply sliding a piece of the waveform for each channel to move the strobe point to the center, and adding up the five copies for the five strobe points in the frame. The SAI has its zero-lag line at the center of the time-interval axis and is truncated at plus and minus $\pm$26.6ms. A value in this region is a common choice for the maximum of the SAI lag axis, as it corresponds approximately to the lower frequency limit (maximum period) of human pitch perception.

The processes of STI and autocorrelation both produce an image that is strongly influenced by the pitch of the sound. The percept of pitch is determined by the repetition rate of a sound waveform, (the pulse rate in a pulse-resonance sound) with faster repetition rates corresponding to higher pitches. STI gives a strong peak in the output function at a lag corresponding to the repetition interval of the waveform (since the signal is well-correlated with itself at this point). The SAI is expected to be stable when a sound is perceived as stable by a human listener. Sounds with a repetition rate of above about 60Hz are perceived as stable tones, with pitch corresponding to the repetition rate. As this rate is reduced, the pitch percept gradually disappears and the individual cycles of the input signal begin to separate out. The lower limit of pitch perception is around 30Hz (Krumbholz et al., 2000). With a lag dimension extending to 26.6ms, sounds with a repetition rate of above about 38Hz will lead to a stable vertical ridge in the image, thus providing an a good approximation to the limits of human perception.

C Details of rectangle selection

To represent each SAI using a sparse code, we first defined a set of local rectangular patches, that covered the SAI. These Rectangles are used to identify patterns that are local to some part of the SAI, and they have different sizes in order to capture information at multiple scales. This approach is modeled on methods that have been used for image retrieval, involving various kinds of multi-scale local patterns as image feature vectors.

We have experimented with several schemes for selecting local rectangles, and the specific method that we used is based on defining a series of rectangles whose sizes are repeatedly doubled. For instance, we defined baseline rectangles of size 16×32, then multiplied each dimension by powers of two, up to the largest size that fits in an SAI frame.

In one group of experiments we varied the details of the box-cutting step. In our baseline we use rectangles of size 16×32 and larger, each dimension being multiplied by powers of two, up to the largest size that fits in an SAI frame. We varied the base size of the rectangle, starting from the sizes 8×16 and 32×64. We also restricted the number of sizes, by limiting the doublings of each dimension. This restriction serves to exclude the global features that are taken across a large part of the auditory image frame. In a separate series of experiments we instead started from a rectangle size equal to the dimensions of the SAI frame, working downwards by repeatedly cutting the horizontal and vertical dimensions in half. This set excludes features that are very

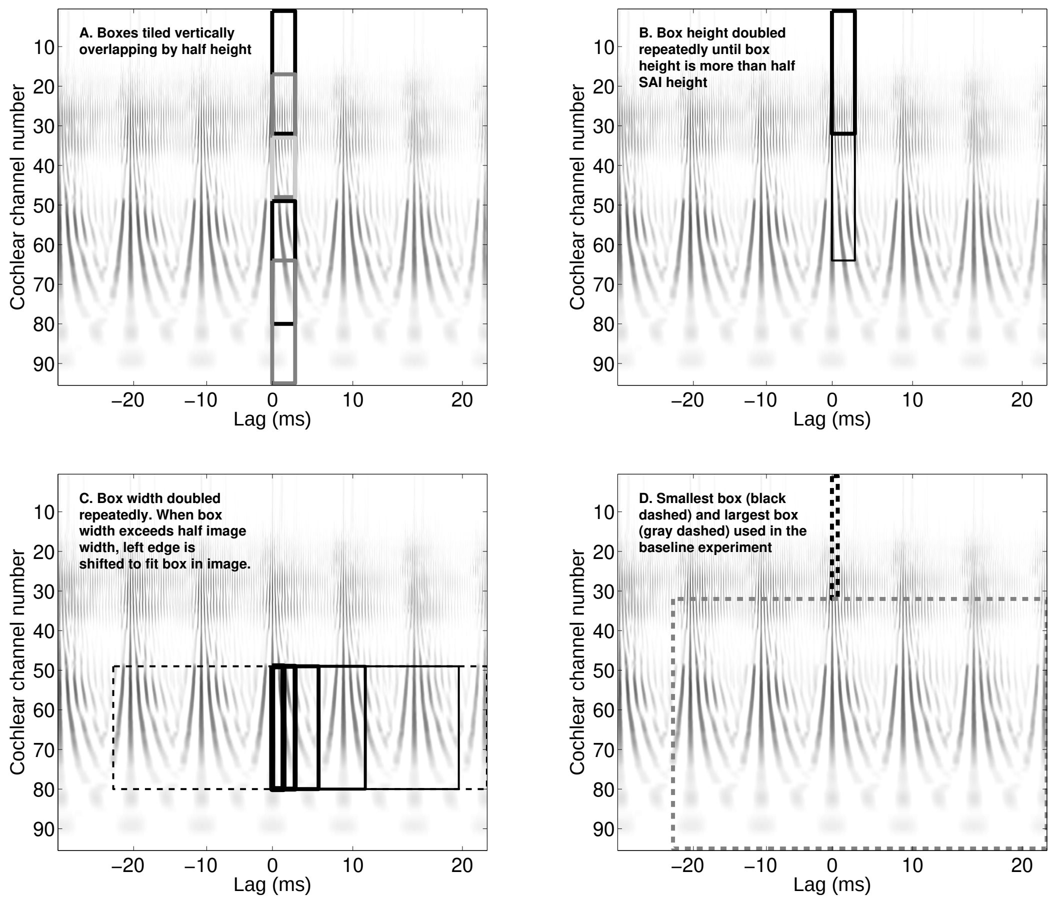

Figure 12: Defining a set of local rectangle regions. Rectangles are chosen to have different sizes, to capture multi-scale patterns. See appendix A. In the default set of parameters we used, the smallest rectangle is 16 samples in the lag dimension and 32 channels high, and the largest is 1024 samples by 64 channels.

local in the auditory image. While the codebook sizes remained fixed at 256, the total number of feature dimensions varied, proportional to the number of boxes used, and performance within each series was found to be monotonic with the total number of feature dimensions. The complete set of experimental parameters are shown in Table 4.

Given the set of rectangles, we create dense features from each rectangle. The image inside the rectangle is down sampled to the size of the smallest box (16 $\times$ 32 with the default parameters). The effect of this rescaling is that large rectangles are viewed at a coarser resolution. To further reduce the dimensionality of the data we compute the horizontal and vertical marginals (the average values for each column and row in the rectangle), and concatenate the two vectors into a single real-valued vector per rectangle. In the default case, this approach leaves a $16 + 32 = 48$ element dense feature vector for each rectangle.

This multi-scale feature-extraction approach is a way to reduce the very high dimensional SAI space to set of lower-dimensionality local features at different scales, as a step toward making sparse features. Different box sizes and shapes capture both the large-scale image structure, corresponding to pitch and temporal coherence, and the microstructure corresponding to the resonances following each pulse. Wide boxes capture long-term temporal patterns; a smaller height on these restricts the temporal pattern features to a localized frequency region and captures local spectral shape. Tall boxes capture overall spectral shape; smaller widths on these include different scales of temporal pattern with the spectral pattern. Intermediate sizes and shapes capture a variety of localized features, such that even when multiple sounds are present, some of the features corresponding to regions of the SAI dominated by one sound or the other will often still show a recognizable pattern. The use of the marginals of each box reduces the dimensionality into the following sparse-code extraction step, while preserving much of the important information about spectral and temporal structure; even with this reduction, the dimensionality into the sparse code extractors is high, for example 48 with the default parameters.

D Summary of Experiments

Table 4: Parameters used for the SAI experiments

Parameter Set	Smallest Box	Total Boxes	Means Per Box	VQ MP	Box Cutting
Default “baseline”	32×16	49	256	VQ	Up
Codebook Sizes	32×16	49	4, 16, 64, 256, 512, 1024, 2048, 3000, 4000 6000 8000	VQ	Up
Matching Pursuit	32×16	49	4, 16, 64, 256, 1024, 2048, 3000	MP	Up
Box Sizes (Down)	16×8 32×16 64×32	1, 8, 33, 44, 66 8, 12, 20, 24 1, 2, 3, 4, 5, 6	256	VQ	Down
Box Sizes (Up)	16×8 32×16 64×32	32, 54, 72, 90, 108 5, 14, 28, 35, 42 2, 4, 6, 10, 12	256	VQ	Up