MUS422/EE367C Class Project, March 15, 2000

Need-Adaptive Variable-Bitrate Audio Encoding

Gunnlaugur Thor Briem

Stanford University, Department of Electrical Engineering

Abstract— **Most perceptual audio codecs in current use are designed to adhere to a fixed bitrate restriction, and thus assign the same number of bits to all frames, optimizing bit allocation only among the frequency components within each frame. This causes no problems when encoding audio samples whose perceptual entropy is fairly constant over time, but in the case of nonuniform required coding effort (especially in the presence of transients), this approach wastes bits where they are less needed, while starving the signal components that need those bits. This report describes an audio codec, GPAC, whose bit allocation scheme attempts to conform more closely to the time-varying bitrate requirement, while satisfying a hard maximum-average-bitrate constraint over a larger time horizon. The resulting codec shows improved performance in handling transients at relatively low bitrates, while not overly degrading the fidelity of non-transient segments.**

1 Introduction

Variable-bitrate audio coding aims to optimize the allocation of bits not only across frequency components within each time frame, but also across different time frames, in order to exploit nonuniform distribution of "bit need" over time in the source sample. The most common cause of locally heightened bit need is the presence of a transient, which implies that a signal with significant transients might benefit from variable-bitrate coding. The following section explores this in more detail.

The remaining sections describe the design and implementation of GPAC (Gulli's Perceptual Audio Codec), a straightforward TDAC transform-based audio coder armed with a variable-bitrate bit allocation procedure. First the overall structure of the coder is presented, then the individual components are discussed, with an emphasis on the bit allocation algorithm, and lastly the performance is evaluated and discussed.

2 Transients and pre-echo

The TDAC transform does perform perfect reconstruction in the absence of quantization noise, but in practice we'll always introduce quantization noise in the coding process. Thus the degree to which perfect reconstruction fails under quantization is a major factor in understanding aberrations and artifacts experienced by transform-based perceptual audio coders. It turns out that one of the major artifacts occurs when the assumption of stationarity fails, i.e. when transients are present. This artifact is pre-echo, the temporal spreading of quantization error due to a transient, over the whole block the transient occurs in.

The larger the transform block size, the more likely the listener is to notice this premature echo of the transient, and thus the more perceptually disturbing this effect will be. Therefore, high-quality codecs have approached this problem by using a variable-sized transform block, shortening it when a transient is detected, to better localize the spreading of the quantization error.

That approach works, but has some shortcomings:

- It complicates the encoder and decoder design. Other components must account for different block sizes: psychoacoustic models must be adapted for different samplings of the frequency spectrum, and the bitstream encoding must express the block size as well.

- It is binary ... either a transient is detected or it isn't, so either the shorter block size is chosen or it isn't [1]

- Shorter transform blocks exhibit inferior frequency resolution, thus decreasing the effectiveness of bit allocation by psychoacoustic modeling.

Instead of improving noise localization by shortening the block size, another approach is to *decrease* the quantization noise in blocks where transients are detected. This can be done without violating the average bitrate constraint, by "stealing" bits from other non-transient blocks, and adding them to the block containing the transient. This does not appreciably complicate the design (modifications are required only in the bit allocation component of the encoder, and the decoder doesn't even need to know) is continuously scalable (assuming a continuous, i.e. non-discrete evaluation of the degree of transient presence), and maintains frequency resolution.

A possible disadvantage of this approach is degradation of quality in the blocks from which bits are "stolen". The importance of this disadvantage depends on the accuracy of the evaluation of "bit need".

It turns out that a good indicator of whether a transient is present in a given block or not, is bit need. Where bit need is significantly higher than average, this usually signifies the presence of a transient. This is often used as a criterion for determining transient presence in deciding whether to shorten the transform block. For the purposes of the approach described here, bit need can be used as a *continuous* measure of the *degree* of transient presence, and hence also as a non-discrete determinant of the number of bits to allocate to a given block.

The GPAC codec uses such an approach.

3 Codec structure

The structure of the codec is an entirely standard one: a time-frequency transform, bit allocation, FP quantization and bitstream formatter. This is illustrated in fig. 1, along with the relevant control signal dependencies.

[1] granted, multiple block sizes could be provided for, but this would further aggravate the aforementioned

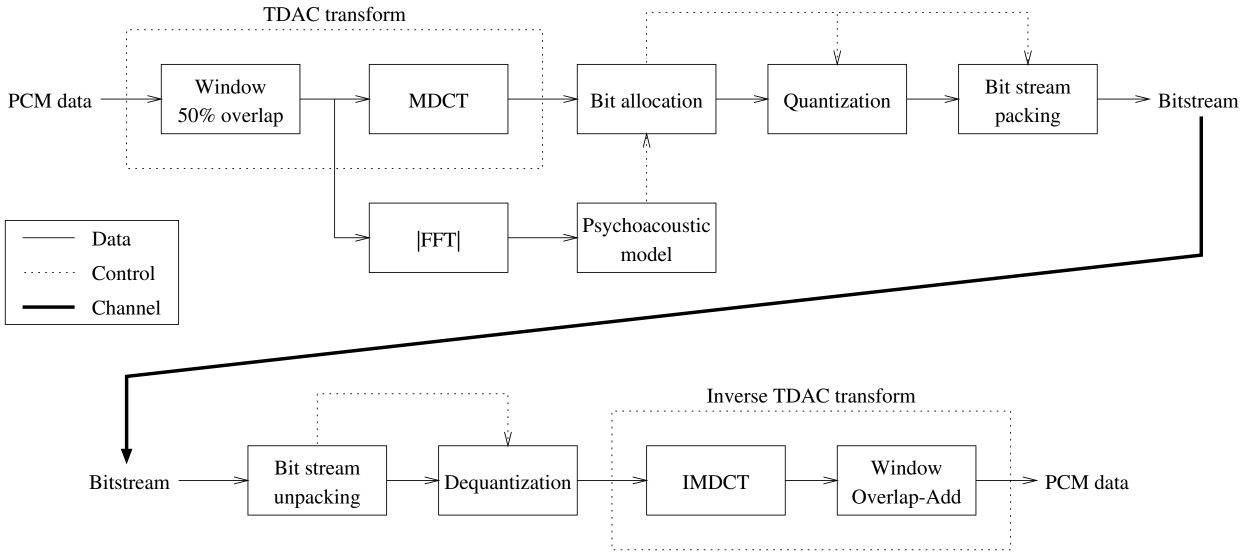

Figure 1: Structure of GPAC encoder and decoder

3.1 The TDAC transform

The time-frequency mapping in this coder is an oddly-stacked time-domain aliasing cancellation (TDAC) transform that gives perfect reconstruction in the absence of quantization. The TDAC transform consists of a series of modified discrete cosine transforms (MDCT) performed on 50%-overlapped blocks windowed such that the squared window functions overlap-add to a constant 1. Only the lower half of the frequency-domain output is maintained, so 50% overlap is achieved without increasing the data rate.

The particular window function used here is a Kaiser-Bessel-derived window with $\alpha = 4$, and the transform is performed on blocks of fixed length, 2048 samples, resulting in blocks of 1024 frequency lines.

3.2 Psychoacoustic model

A psychoacoustic model is used to determine the coding accuracy needed for each Bark band across multiple time frames, based on an estimate of the spectral envelope of the sample. A masking curve is constructed within each frame, depicting the minimum signal power that would be perceived by the human auditory system. This curve is estimated as a function of the spectral envelope of the incoming signal, by superimposing individual masking curves due to individual peaks detected in the spectrum, onto the threshold of hearing in quiet.

A very simple piecewise linear (in the dB-vs-Bark scale) masking curve model was used. It is a triangle around the peak, with slope 27dB/bark on the left side and $-27 + 0.37(L_m - 40dB)$ on the right side, where L_m is the signal power of the masker peak. Peaks are detected by a heuristic criterion: if a spectral line magnitude is larger than both its nearest neighbors, and larger than 70% of the sum of its four nearest neighbors, and larger than 35dB, it is regarded as a masker peak.

increase in codec design complexity, and has not been done in current high-quality codecs

The individual masker curves are added together on the log scale to produce the overall masking curve M_k (where k is an index into the frequency bins).

3.3 Bit allocation

In bit allocation we try to shape the quantization noise so that it falls below the masking curve and is thus inaudible. This means we want to allocate mantissa bits (for the floating-point quantization that follows) so as to maximize not the SNR, but the sum (in logarithmic measure, product in linear measure) of the SNR and the masking curve, i.e. SNR-SMR, where SMR is the signal-to-masking curve ratio. This is intuitively reasonable: where the masker is high, the SNR doesn't need to be as high.

Using the mathematical representation of floating-point quantization noise,

$$\langle q^2 \rangle = \frac{\langle x^2 \rangle}{3 \cdot 2^{2R}}$$

this is posed as an optimization problem:

$$\text{Minimize} \quad \frac{1}{N} \sum_{k=0}^{N-1} \frac{(x_k/M_k)^2}{3 \cdot 2^{2R_k}} \text{subjectto} \frac{1}{N} \sum_{k=0}^{N-1} R_k = R$$

where R is the average bitrate constraint, and R_k are the optimal bit allocations within the frame, k again being a frequency bin index. The solution is

$$R_k = R + \log_2 \left| \frac{x_k}{M_k} \right| - \text{SFM}$$

where the spectral flatness measure SFM is defined as

$$\frac{1}{N} \sum_{m=0}^{N-1} \log_2 \left| \frac{x_k}{M_k} \right|$$

In this coder, this result is used as an initialization to an iterative procedure that attempts at each step to quench the highest remaining bit need of any Bark band in all the frames in the optimization horizon[2]. The bit need R_{fb} (where b is a Bark band index and f is a frame index) is simply the sum of the bit needs R_k of the band's constituent frequency lines. Bits are initially allocated to each Bark band as $B_{fb} = n_b \lfloor R_{fb}/n_b \rfloor$ where B_{fb} is the total number of bits allocated to Bark band b in frame f and n_b is the number of linear frequency lines in band b. The remainder r_{fb} is defined as the difference $B_{fb} - R_{fb}$.

The optimization then proceeds in two phases. First, in case too many bits were allocated in the above initialization, bits are stolen until the average bitrate constraint R is satisfied by the sum of all R_{fb}. Then bits are given back until the constraint is satisfied as exactly as possible. This converges to within 2-3 bits of the exact constraint.

[2]In the current implementation of the coder, this horizon is infinite, i.e. the entire sample is optimized at once. This was acceptable in a prototype system since the test input was not too large to fit in memory on a normal PC workstation. In a practical system whose input is arbitrarily large, a maximum horizon would need to be established for the optimization.

In the first phase, the number of bits left for allocation is negative. At each step the optimizer finds the smallest r_{fb} among bands that don't already have 0 bits allocated to them, reduces the corresponding B_{fb} by n_b and increase the r_{fb}, and increases its tally of bits left by n_b bits.

In the second phase, the number of bits left for allocation is positive. At each step the optimizer finds the largest r_{fb} among bands that don't already have 12 bits (the maximum) allocated per spectral line, and that aren't larger than the number of bits left. It increases the corresponding R_{fb} by n_b, decreases the r_{fb} and decrements the number of bits left.

This algorithm always terminates, but the implementation was initially plagued by the fact that the search took an inordinately long time. Sorting the frames and bins by r_{fb} was not practical since this ordering is changed at each step. Instead, the speed problem was ameliorated by encapsulating the semi-ordering of r_{fb} in what I call a double-ended minmax priority queue. This is a somewhat convoluted data structure, based on the normal pseudo-ordered priority queue concept, but extended in such a way as to enable insertion and extraction of both the maximum *and* the minimum element in logarithmic time. In practice, this turned bit allocation from being by far the dominant factor in encoding (making encoding take many times the timelength of the sample on a normal PC), to being completely negligible compared to the time taken by the FFT and MDCT. A representative speed comparison was made: The original encoding procedure with a search-based bit allocation routine took 18 minutes on a workstation PC for a 62-second sample. With the priority-queue-based approach, the encoding takes 19.5 seconds, almost all of which is spent performing the FFT and MDCT.

There are interesting extensions and variations to try out on this scheme, which are discussed in the Conclusions section.

3.4 Quantization

A non-uniform FP quantization is employed, with a fixed scaleword size of 4 bits, but a single scaleword is assigned to all frequency lines in a given Bark band. Also constant across each Bark band is the mantissa size, determined by the bit allocation procedure described above.

3.5 Bitstream packing/unpacking

The bitstream format is comprised of a 20-byte header identifying the file as being GPAC format, and describing the bitrate and sample rate, followed by the variable-length frames. The frame format is designed for a perfect channel (such as a file storage device or a "trusted" communications system with sufficient error correction), and is described in the schema below.

Bitsize	Data
4	Frame-wide scale factor
4	Mantissa size m_0 for Bark band 0
0/4	Scale factor for Bark band 0 (omitted if mantissa size is 0)
m_0	Mantissa for first spectral line of bark band 0
m_0	Mantissa for second spectral line of bark band 0
$\vdots$	$\vdots$
4	Mantissa size m_0 for Bark band 1
0/4	Scale factor for Bark band 0 (omitted if mantissa size is 1)
m_0	Mantissa for first spectral line of bark band 1
$\vdots$	$\vdots$

Packing into and unpacking out of this format is a matter of simple data logistics, and will not be discussed here.

4 Performance

Representative samples of codec output can be found at

`http://www-ccrma.stanford.edu/~gthb/demo/`

The ITU-R 5-point scale scores reported here should not be taken as reliable, since they are made by the author, who is by no means an experienced listener or evaluator of audio fidelity, and obtained using inexpensive standard PC audio equipment in a non-isolated sound environment. Rather than repeat that multiple times below, it is stressed here that these are very subjective informal evaluations and should be viewed as such.

At 128 kb/sec, the codec is transparent for all the samples considered. At 96kb/sec some of them display slight artifacts, dropping to 4.5 or 4.0 on the ITU-R 5-point scale. At 64 kb/sec, most of the samples are "okay" (3.5), but one exhibits glaring artifacts which drop it to 2.5. At lower bitrates the codec can be said to break down entirely, although some samples don't quite go down to 1.0 at 48 kb/sec.

The main artifact noticed at the higher bitrates is warbling at medium-high frequencies, caused by the removal of bits from steady-state frames. This is most pronounced in the `isdeewane` sample, which has a rich harmonic background sound that suffers at the hands of the codec.

5 Conclusions and future work

The codec performs surprisingly well [3] at bitrates down to 96 kb/sec, which inspires some confidence in the bit allocation scheme described. The major artifacts can, I believe, be ascribed to imperfect evaluation of the bit need at each frame/band position in the sample. An interesting future extension will therefore be to find a more advanced way of evaluating this important metric.

Other interesting ideas for future work include morphological filtering of the bit allocations along the time dimension, to avoid frequent changes in the accuracy of a given frequency band (because such changes may be more disturbing to the listener than simply staying at the lower accuracy), the inclusion of temporal masking, and a variable window shape (sine or KBD) depending on presence of close spectral peaks in the sample. All of these modifications are straightforward in the present design, since the bit allocation is already occurring over an arbitrary number of frames in the temporal dimension.

[3] certainly better than the (inexperienced) author expected