```
_____________________________________
___    |__    |__    |__    |_  ____/
__  /| |_  /| |_  /| |_  /| |  /
_  ___ |  ___ |  ___ |  ___ / /___
/_/  |_/_/  |_/_/  |_/_/  |_\____/
```

All-American-Advanced-Audio-Codec

Tim O'Brien Jennifer Hsu
Colin Sullivan Mayank Sanganeria

Perceptual Audio Coding
MUSIC 422 / EE 367C

Center for Computer Research
in
Music and Acoustics

Stanford University

1 Project Overview

In AAAAC, we aim to deliver perceptually lossless audio quality at minimal data rates. Rather than design for low latency/streaming applications, we take an approach that assumes encoding delay is acceptable given the realizable gains in audio quality. In addition to the baseline MDCT-domain audio code, we implement Huffman coding, stereo coding, block switching, and a variable bit rate. These are discussed in more detail below.

2 Huffman Coding

Huffman coding is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol. Huffman coding gets better the more unbalanced the probabilities are.

2.1 Motivation

Huffman coding is a lossless compression technique and therefore it does not affect the output quality of the coder. As long as the gains in data transmission are greater than the overhead of including information about the Huffman coding tables, using Huffman makes the coder better. Though Huffman is not the only form of lossless compression, it is popular among audio coders which means it has been tried and tested successfully in the past.

2.2 Methodology

As is done in MPEG 1 Layer 3, the frame of data containing the mantissas is split up into 3 regions. The first region contains the low frequency data, which typically consists of various values with no significant pattern. The second region contains just 1 and 0. The third region has high frequency content and is typically all 0s.

The third region can be coded by simply specifying the length of 0s (or inferring the length from the other 2 regions), while the second region gets gains by collapsing the 16-bit integers representing 0 and 1 to single bit 0 and 1 codes. The first region goes through a more traditional Huffman coding. Currently, the tables that are going to be used in the coder for this region have not been decided.

2.3 Harmony with VBR

The Huffman routine will add the bits saved to the bit reservoir that the VBR routine is using, thus enabling the VBR to work more efficiently by counting the bits saved via Huffman encoding as well.

3 Stereo Coding

Our basic coder codes stereo files in a dual mono process by separately encoding each of the two channels. By taking advantage of the fact that most signals have strong correlations between their left and right channels, we can reduce redundancy by cleverly coding the stereo signals. We implemented Mid/Side (M/S) stereo coding. Instead of transmitting the left and right signals, we transmit their sum and difference signals. We can fully recover the original left and right signals by adding or subtracting the sum and difference signals from each other. In the cases where we have a left channel and right channel that are very similar, we have a small difference signal and use less bits to encode this data.

3.1 M/S and L/R Switching

For every block of an input signal, our codec must decide whether or not to transmit M/S or L/R on a subband by subband basis. We transmit the M/S signals if the following equation is true.

$$\left| \sum_{k=b_{\text{lower}}}^{b_{\text{upper}}} \left(l_k^2 - r_k^2\right) \right| < 0.8 \left| \sum_{k=b_{\text{lower}}}^{b_{\text{upper}}} \left(l_k^2 + r_k^2\right) \right| \tag{1}$$

In this function, l_k indicates kth MDCT frequency line from the left signal and r_k indicates the kth MDCT frequency line from the right signal. b_{lower} and b_{upper} are the lower and upper indices of our subband. This function means that for every subband in our MDCT, if the energy in our side (or difference) signal is sufficiently lower than the energy in our mid (or sum) signal, then we encode the left and right information in this subband to mid and sum information. The 0.8 factor means that if the energy in the side and mid signals differs by 80%, then we switch to M/S mode. This is a tunable parameter, although most of the current literature uses the value of 0.8. Also, note that we used the MDCT frequency lines to make our decision. In many other codecs, the FFT is used.

3.2 M/S Encoding and Decoding

Once we have made the decision to encode a subband of our signal block as M/S, we can calculate our encoded subband information as follows:

$$M_i = \frac{L_i + R_i}{2} \tag{2}$$

$$S_i = \frac{L_i - R_i}{2} \tag{3}$$

In the above functions, L_i and R_i are the left and right samples of our left and right signal data. M_i and S_i are the encoded values. With simple substitutions, we can see that the original left and right channel information can be completely recovered:

$$L_i = M_i + S_i \tag{4}$$

$$R_i = M_i - S_i \tag{5}$$

3.3 Masking Threshold for Stereo Signals

When calculating our masking thresholds, we first calculate the masking curve for the L/R signals in the same way as our original coder. For each masker that we detect, we use the following spreading function to calculate our overall masking threshold for the L/R channels:

$$10\log_{10}(F(z_\triangle, L_M)) = \begin{cases} 0, & \text{for } -\frac{1}{2} \le z_\triangle \le \frac{1}{2} \\ -27(|z_\triangle| - \frac{1}{2}), & \text{for } z_\triangle < -\frac{1}{2} \\ (-27 + 0.367\max(L_M - 40, 0))(|z_\triangle| - \frac{1}{2}), & \text{for } z_\triangle > \frac{1}{2} \end{cases} \tag{6}$$

where $z_\triangle = \text{Bark}(f_{\text{maskee}}) - \text{Bark}(f_{\text{masker}})$ and L_M is the sound pressure level (SPL) of the masker in dB. After we calculate this spreading function, we downshift the function by some amount δ to emulate perceptual masking. We then calculate the mask signal to mask ratio (SMR). Let $\text{SMR}_{\text{L/R}}$ denote the max SMR per band for the left and right channels.

Having calculated the above for the L/R signals, we perform a modified version to find the masking threshold and max SMRs for the M/S signals. We first calculate our basic thresholds on the M/S signals using the same spreading function and downshift as stated above and denote these thresholds as BTHR_M and BTHR_S.

We then calculate the spreading function again on the M/S signals except this time, we do not consider the downshift δ factor. We can call these functions BTHR$\prime_M$ and BTHR$\prime_S$. We then calculate the masking level difference factor (MLD) which gives us another level of detecting noise across the M/S signals. The equation that we used for the MLD is:

$$\text{MLD} = 10^{(1.25(1-\cos(\pi \frac{\min(z,15.5)}{15.5}))-2.5)} \tag{7}$$

for a frequency z in Barks. Please see the attached figure for a plot of the MLD. We incorporate this difference factor by multiplying our basic thresholds for M/S (without the downshift) by the MLD calculated for each frequency of our MDCT lines to get $MLD_M = \text{BTHR}\prime_M$ and $MLD_S = \text{BTHR}\prime_S$.

From here, we can derive our overall masking threshold for each M/S signal as:

$$THR_M = \max(BTHR_M, \min(BTHR_S, MLD_S)) \tag{8}$$

$$THR_S = \max(BTHR_S, \min(BTHR_M, MLD_M)) \tag{9}$$

These equations means that we use the MLD signal whenever there is a chance of stereo unmasking. We calculate our maximum SMRs per band in the same way as we did for the left and right masking thresholds, but just using the overall thresholds for the M/S signals and denote this $\text{SMR}_{M/S}$.

Last, we go through $\text{SMR}_{\text{L/R}}$ and $\text{SMR}_{M/S}$ and we create an overall SMR array. For each band, we check if we will transmit L/R or M/S based on the decision function described above. If we will send the L/R signal for band b, we set SMR[b] = $\text{SMR}_{\text{L/R}}$[b], otherwise, SMR[b] = $\text{SMR}_{M/S}$[b]. This overall SMR array is used for simultaneous bit allocation across both channels for maximum bit savings.

3.4 Some results and thoughts

Please see the appendix for two plots that compare the differences between the MDCT SPL, overall masking threshold, and max SMRs for the L/R signals and M/S signals. The original signal used to generate these blocks had the same data in the left and right channels. Notice that in the M/S plot, we have extremely low SMRs for the difference signal. We will save space by not using bits for that channel. In this case, we would probably use about half the amount of bits that we would use for the L/S signals.

After implementing all the coding, I found that it works well for some signals (castanets, speech, single sinusoidal addition tone), but it does not work well for others (especially the quartet sample). In the bad case, it sounds like the sound is traveling back and forth between the left and right channels. This may be due to a bug in the code.

[1].

4 Block Switching

In order to provide maximum frequency resolution for quasi-stationary signals while minimizing “pre-echo” artifacts related to transients, we use a block switching algorithm similar to MPEG Layer III. Based on a look-ahead transient detection algorithm, short, long, or transition windows are used (see Figs. 1 and 2). To preserve block synchronicity with respect to the stereo channels, multiple short blocks are be implemented in series so that they are equal to one or more long window blocks. Transitions are handled with asymmetric long-to-short (”start”) and short-to-long (”stop”) windows.

We model our transient detection algorithm on the one used in Dolby AC3. That is, we assume that transients are characterized by the sudden presence of broad high-frequency components in the signal spectrum. Thus at each frame, we high-pass the second half of the block and compare the spectrum to a threshold

level (tuned for best results) and to the previous block's spectrum values. If the high-frequency spectrum has sufficient energy and is sufficiently different from previous values, we switch to a block of short windows and pass one bit signifying this change as a control message to the decoder. Similarly, in a block of short windows, we test whether to go back to the long-window, steady-state regime, or to continue on with another block of short windows suitable for a continuing transient. The results of this detection regime for various input audio files are shown in Figs. 3, 4, 5, 6, 7, and 8.

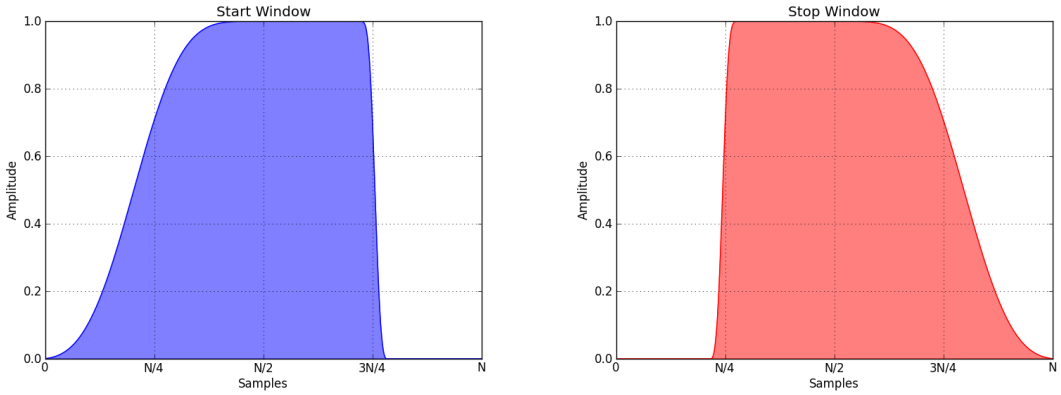

Figure 1: Transition windows used in block switching.

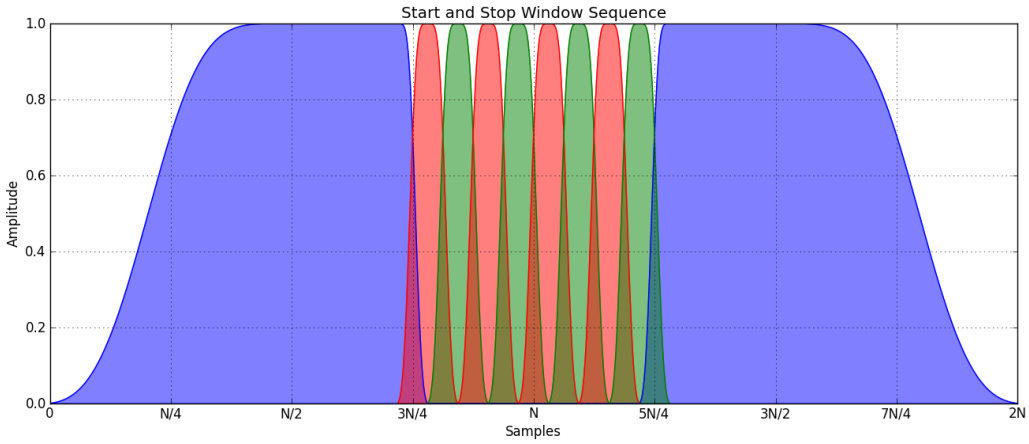

Figure 2: Typical block switching transition to one set of short blocks, then back to the long block length.

More detailed description to come.

5 Variable Bitrate

Our coder now includes a variable data rate coding via a "bit reservoir" similar to MPEG Layer III. This allows the codec to adapt locally to variations in the masking threshold, allocating fewer bits to blocks with heavy masking and utilizing extra bits from the reservoir for more complicated blocks.

Implementing VBR under the current infrastructure was fairly straightforward. Most of the work involved was understanding how the data is being processed and stored in the compressed file. The bit allocation

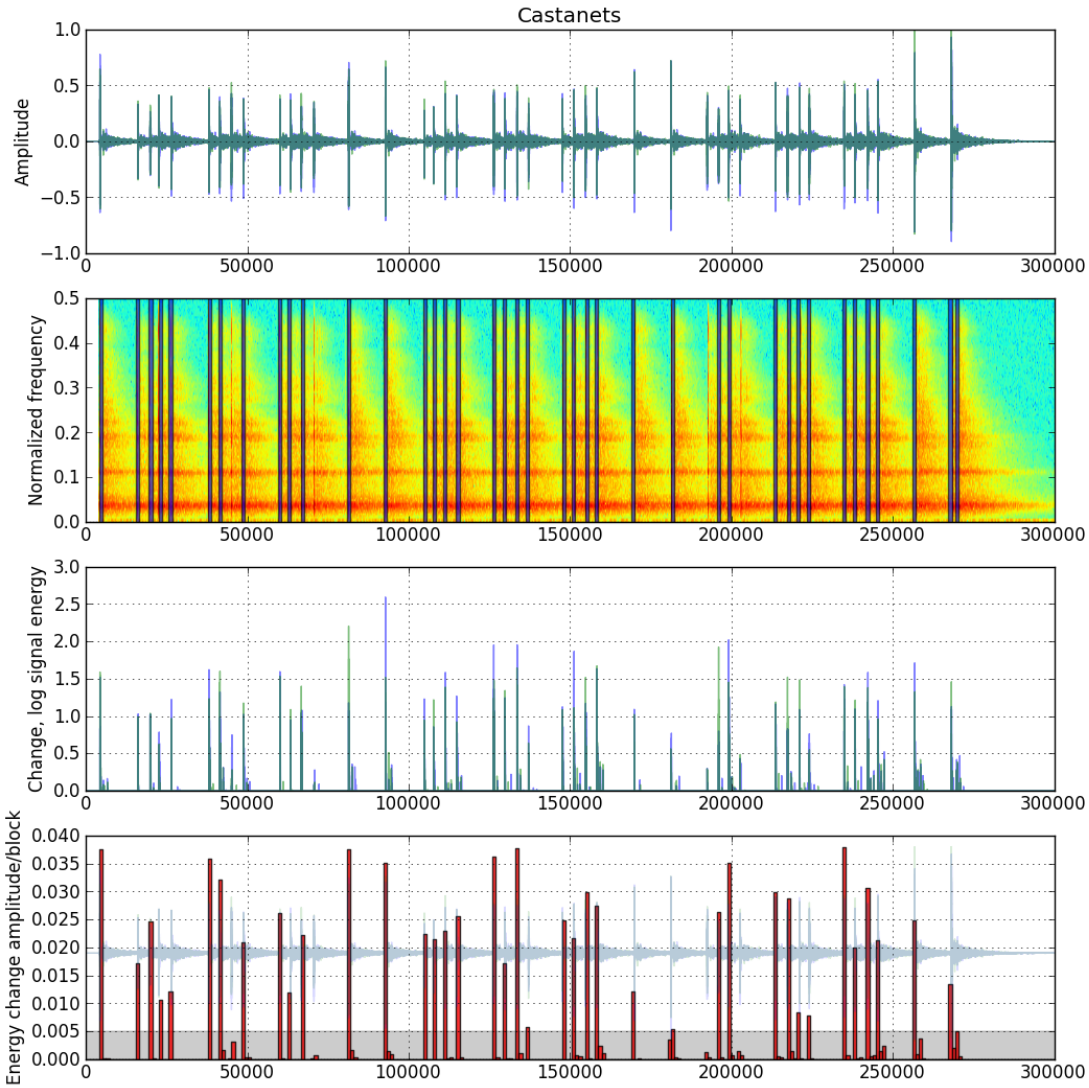

Figure 3: Transient detection in a recording of castanets.

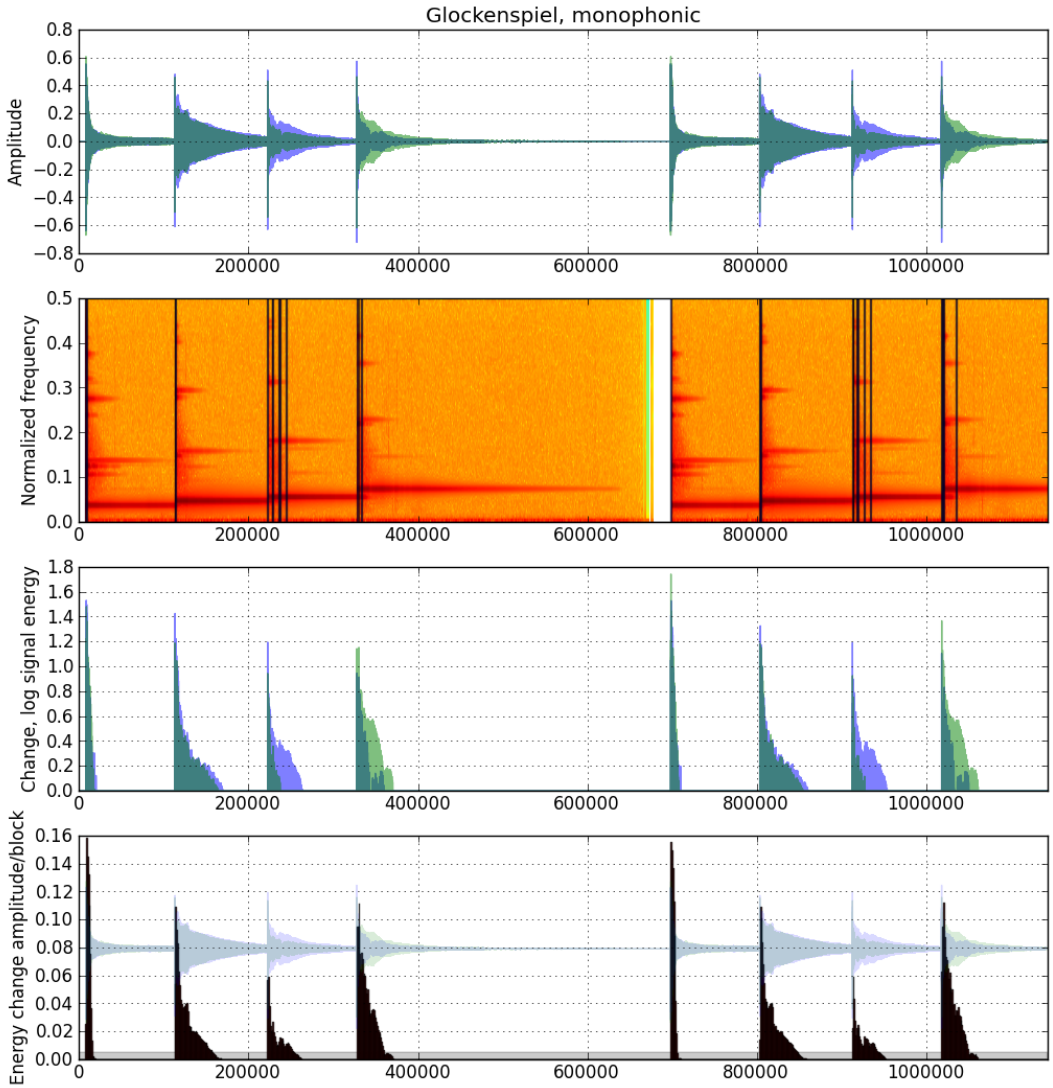

Figure 4: Transient detection in a monophonic recording of a glockenspiel.

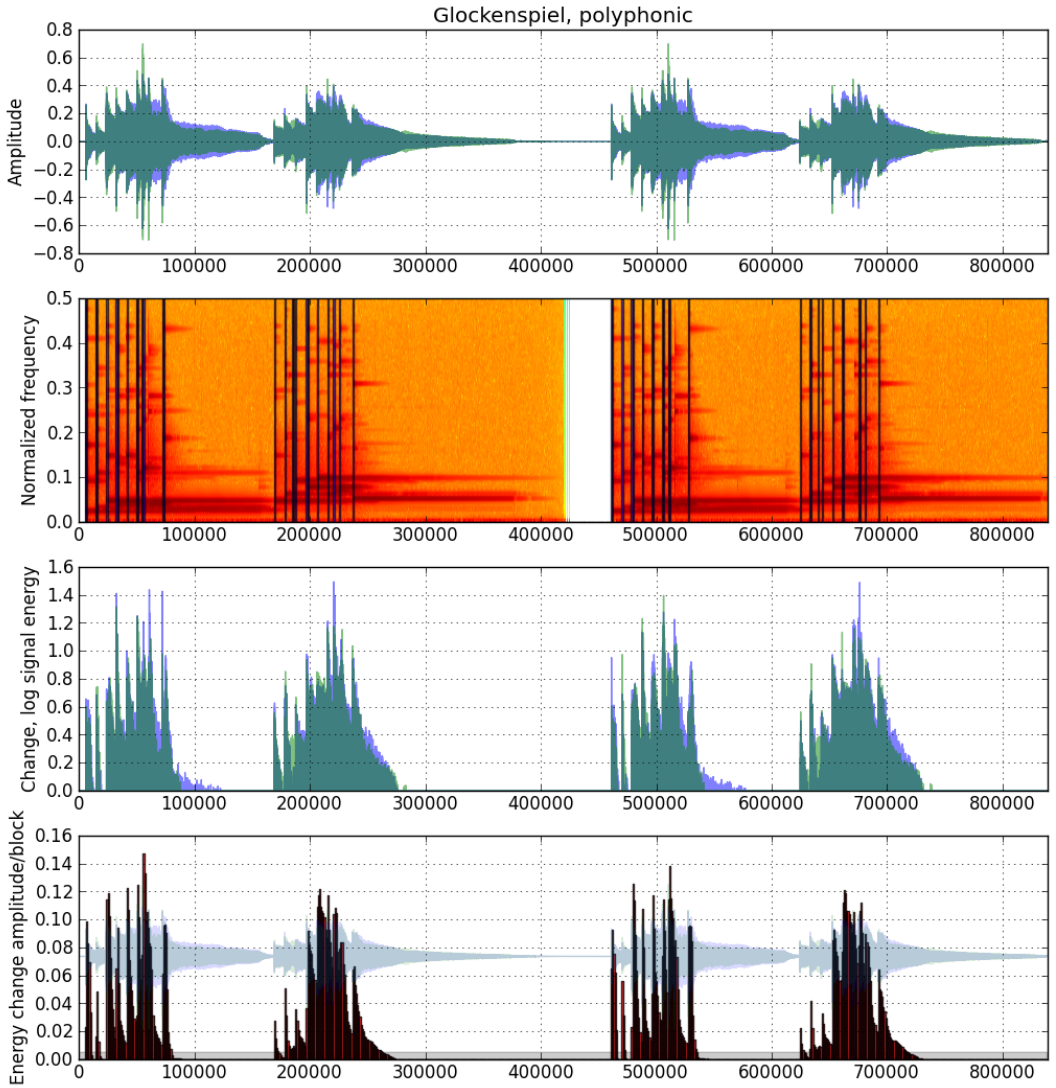

Figure 5: Transient detection in a polyphonic recording of a glockenspiel.

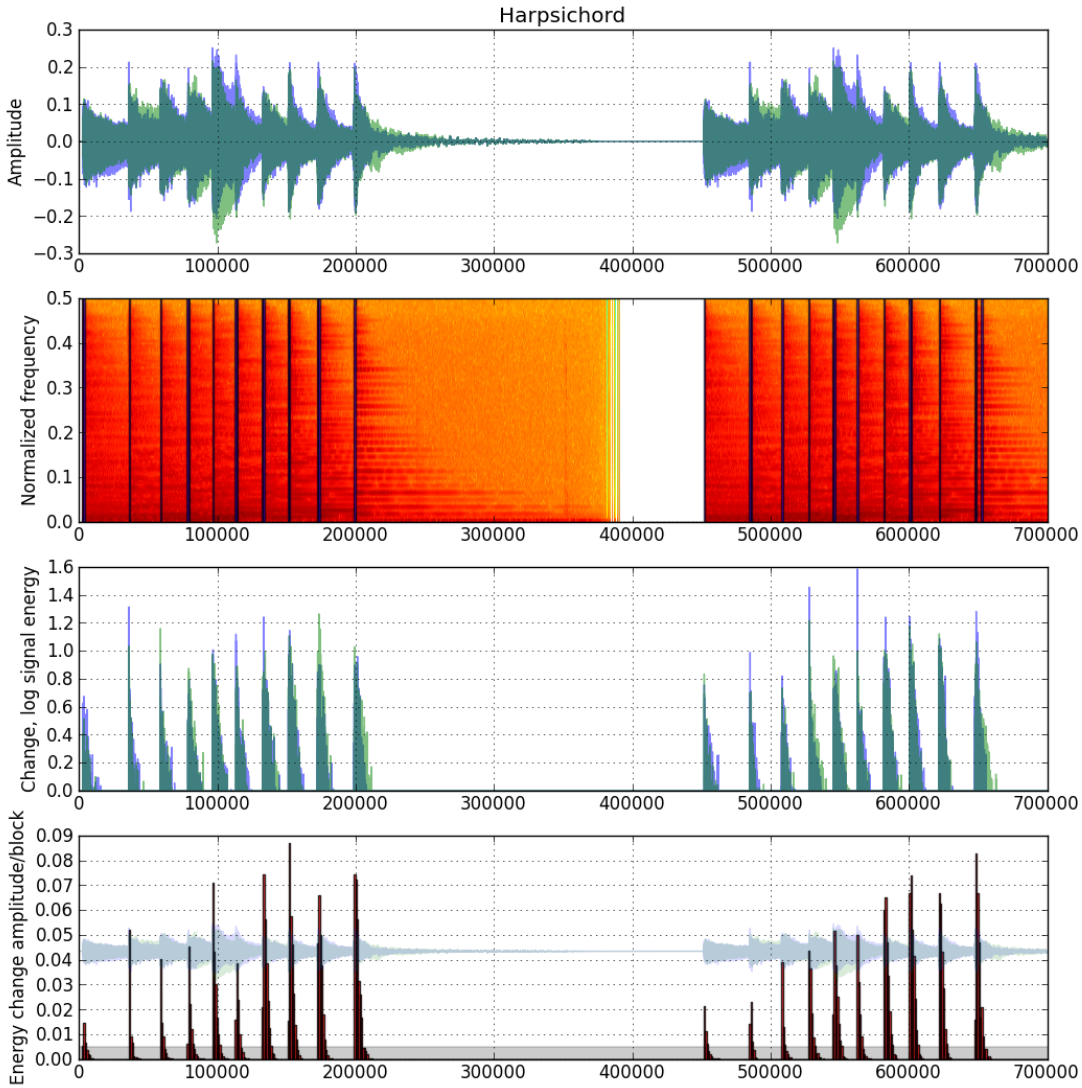

Figure 6: Transient detection in a harpsichord recording.

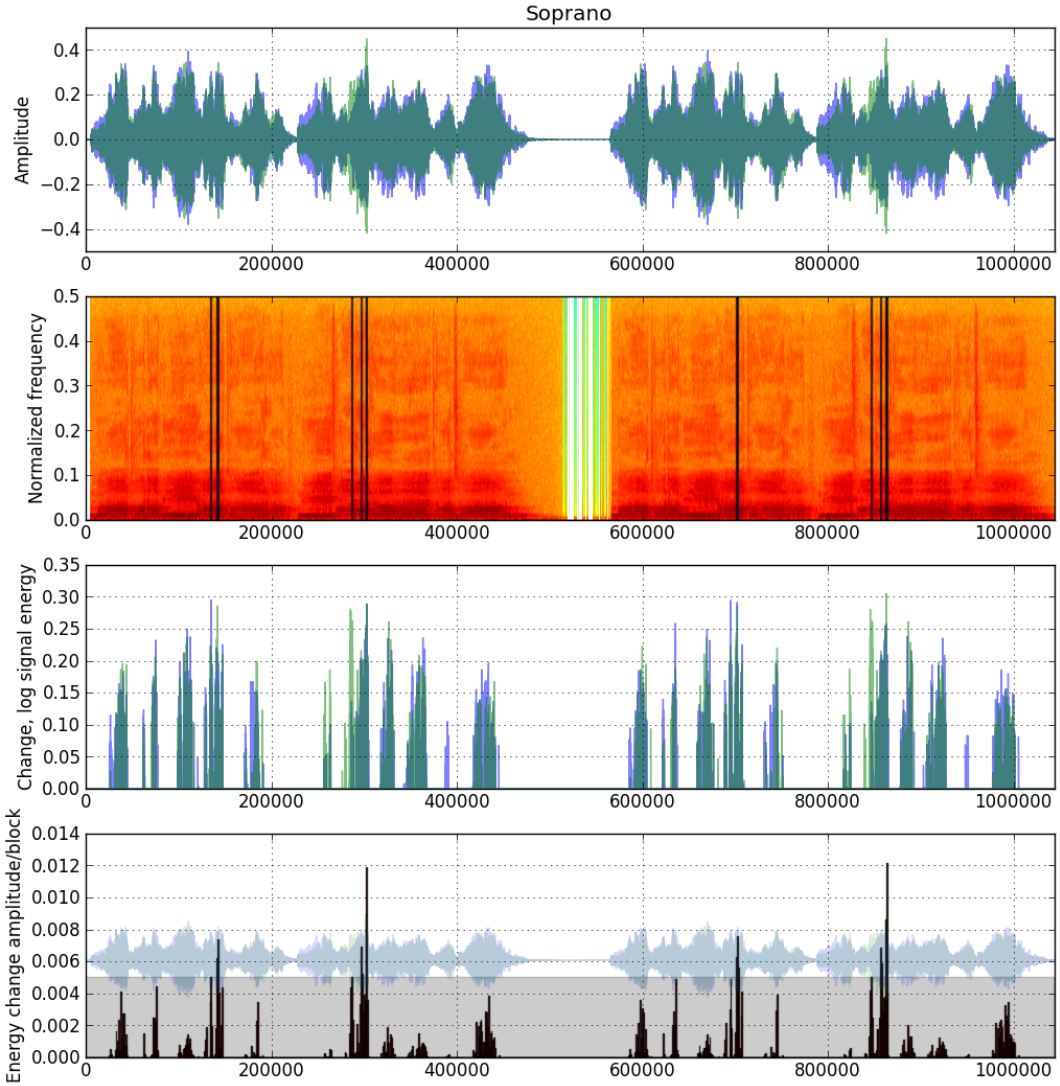

Figure 7: Transient detection in a recording of a soprano.

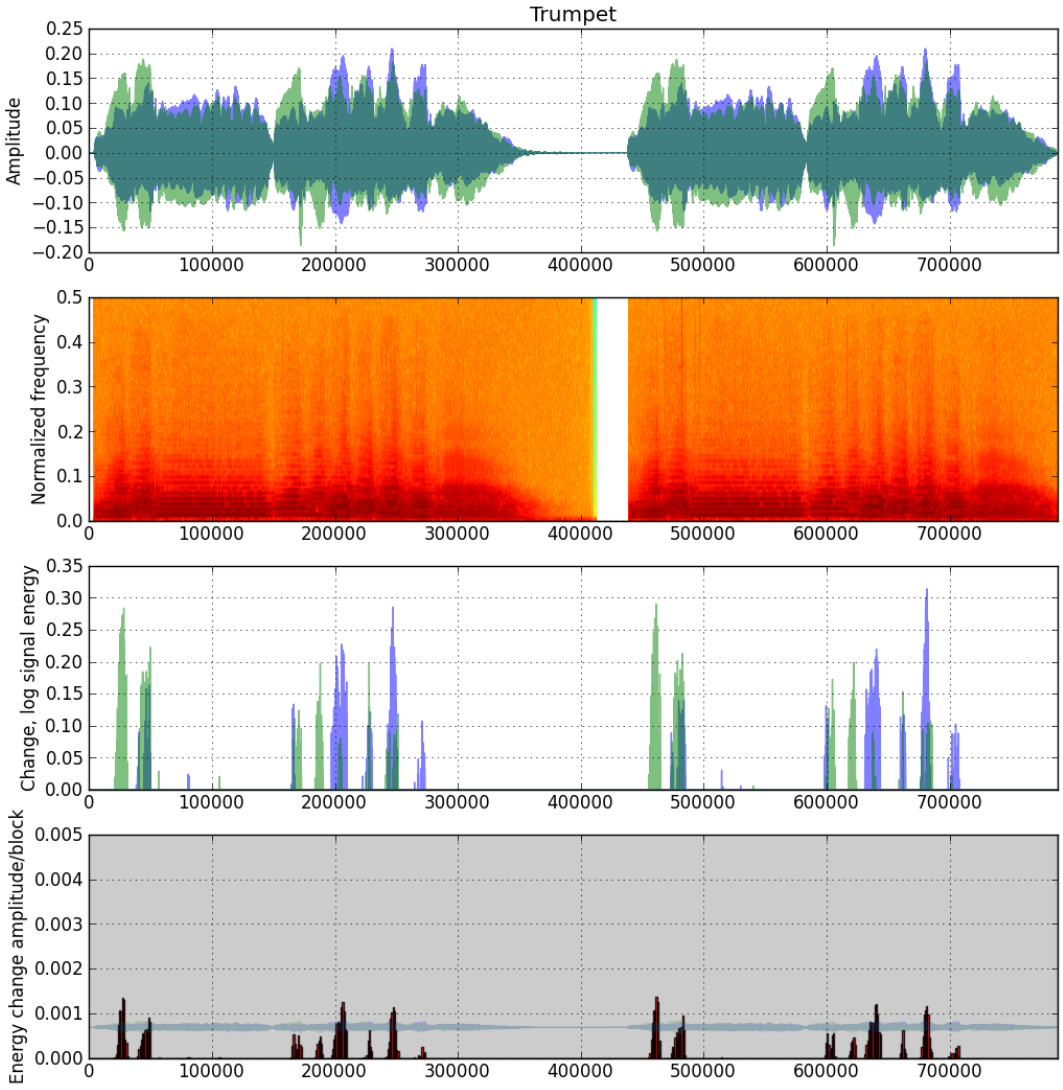

Figure 8: Transient detection in a recording of a trumpet. Note no transients meet the threshold for block switching.

algorithm in our baseline coder was fairly conservative and the variable bit allocation puts the unused bits to good use.

To implement the feature, it was simply a matter of passing the unused bits in the `codingParams` variable so future calls to `bitAlloc` will have these bits added in. This algorithm can certainly be tweaked in the future to utilize the unused bits even more effectively, but since the SMR calculation in the baseline codec is greedy and is merely using only the bits it needs, this technique works out.

Additionally, our codec does not have the same challenges as the bit reservoir described in the book, as we are not concerned with sending frame headers on a regular basis. This means that our blocks can be of varying sizes and be literally stored in the file like that because the headers are just read between each block. This avoids all the issues associated with "pointing" to the location where the mantissas for the current block start.

Figures 12 and 13 show the difference in bit allocation between the baseline coder and our coder with the VBR functionality. As can be seen in the 128 kbps case there is a huge variation in the amount of bits allocated per block and even in many cases less bits than before. In the 256 kbps case, because there are more bits in the original budget, the plot is more flat because the bits in the reservoir were used less.

6 Results

An coded test file at 128 kbps and 256 kbps can be found in `quar_decoded_128.wav` and `quar_decoded_256.wav`. The compressed versions are included as well. Our compression ratio is still as expected, ≈ 5.42 for 128 kbps and ≈ 2.8 for 256 kbps. As mentioned in section 3.4, there are some stereo artifacts that we have yet to fix. These audio samples were encoded with only the VBR and stereo features of our codec. We are still working on integrating the Huffman encoding and block switching logic into our codec.

7 Future Work

How we might improve the coder going forward.

References

[1] J. Johnston and A. Ferreira, "Sum-difference stereo transform coding," in *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, vol. 2, pp. 569 –572 vol.2, mar 1992.

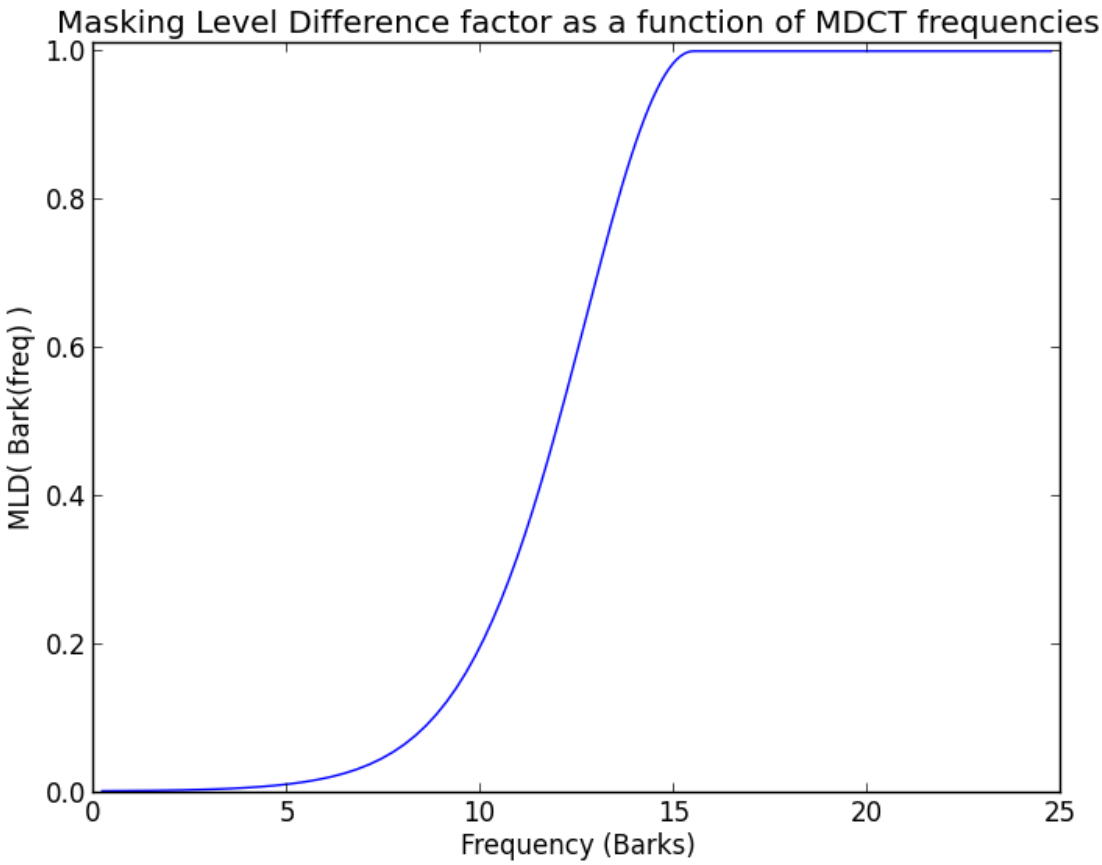

Figure 9: Masking level difference factor for Bark frequencies.

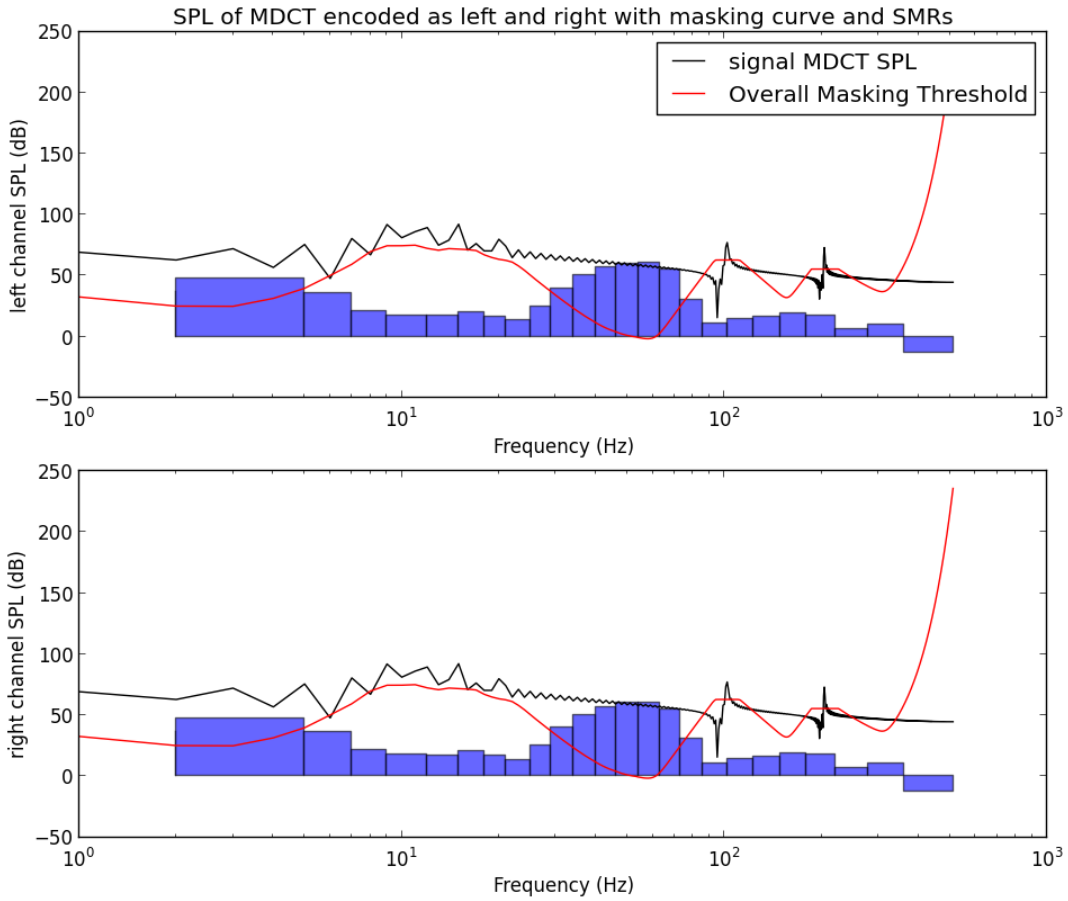

Figure 10: Left and right channels: MDCT SPL, overall masking threshold, and max SMRs.

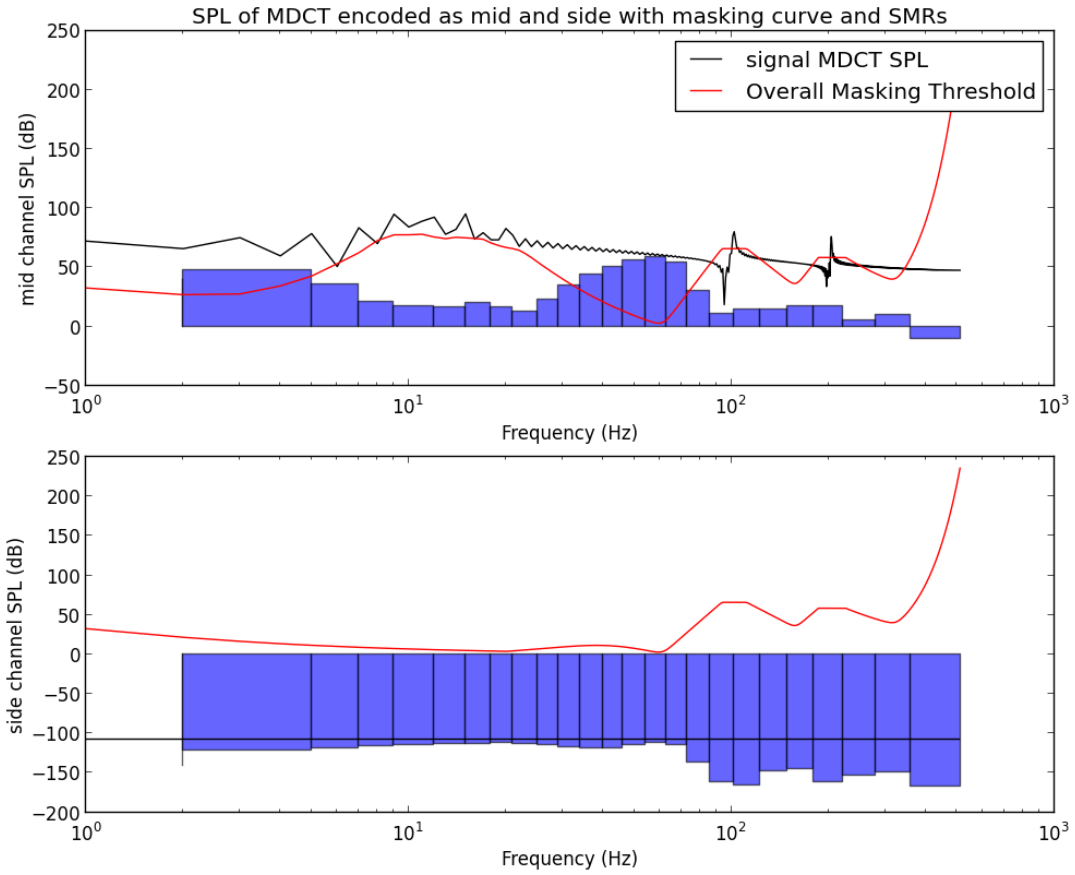

Figure 11: Mid and Side channels: MDCT SPL, overall masking threshold, and max SMRs.

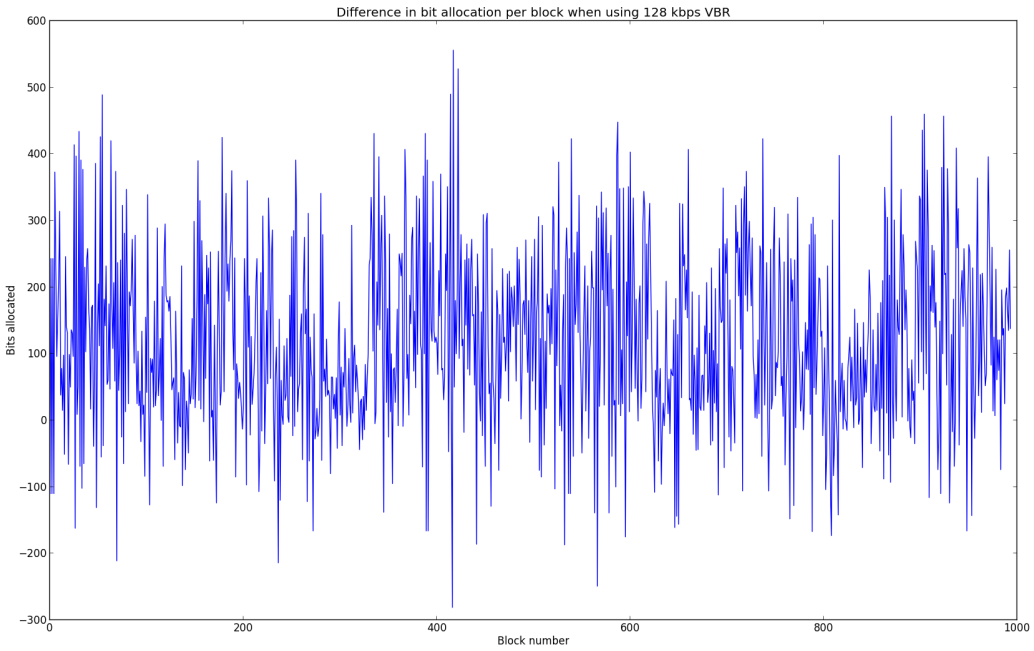

Figure 12: Difference in bit allocation for 128 kbps (aaaac - baseline)

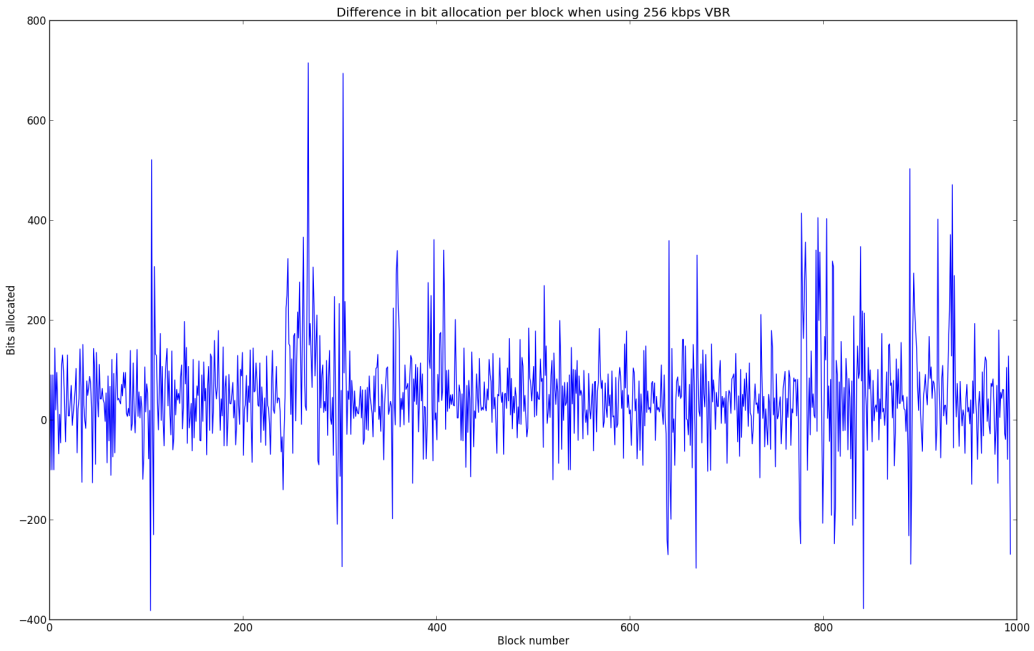

Figure 13: Difference in bit allocation for 256 kbps (aaaac - baseline)