Vorbis

The_Nihilant

GPN12

June 10, 2012

Vorbis

Vorbis I (from now on just Vorbis) is a good quality and performance lossy general purpose audio codec.

An audio format is a way to digitally represent a sound or an audio signal. Some general properties of digital sound representation are:

- the bitrate, that is the number of bits per second needed to represent the signal
- number of channels (2 in stereo representations)
- sampling frequency

Other parameters depend on the specific format considered.

The first audio formats were not compressed. That means that data was taken directly from the input, digitized and stored without further processing. Uncompressed formats were there since the beginning of digital audio processing, when computing power was scarce. Typically uncompressed formats store separately every channel in a PCM format.

PCM formats (Pulse Code Modulation) are very simple, and consist in sampling the input signal at regular intervals and quantizing the value using finite-precision machine numbers. The various formats differ for 4 different characteristics:

PCM

- Sampling frequency: number of samples per second.
- Number of channels, typically 1 (mono) or 2 (stereo).
- Sample size: how many possible values for each sample.
- Representation of the sample: usually integers (with or without sign), but floating point formats exist.

Sampling example

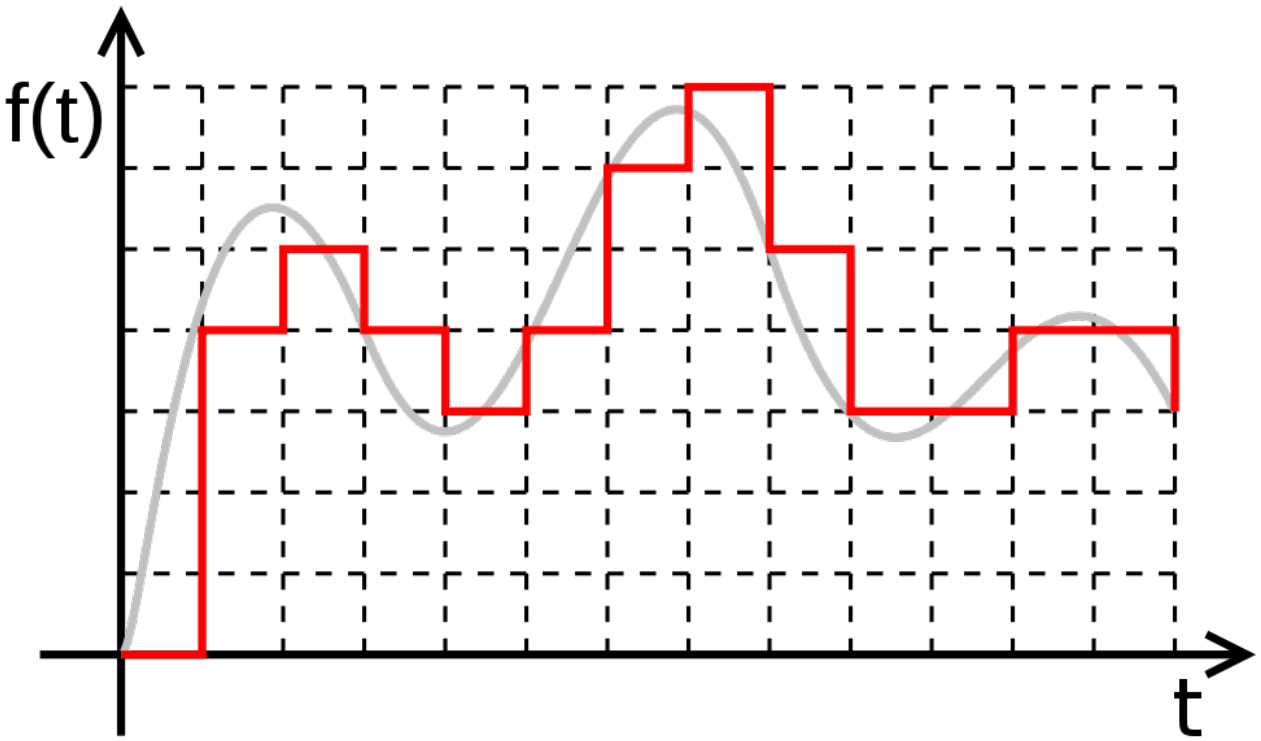


Figure: Example of sampling and quantization of a signal (in red) in a PCM format.

An audio file in a PCM format contains a long series of numbers that represent the shape of the signal in any given sampling point. The CD Audio format (Redbook) is a 16-bit signed 44100Hz Stereo PCM format. This means that 44100 samples per second are gathered, and that the precision is 16 bit (that is, the intensity of the signal can oscillate between 2^{16} possible values).

Sampling errors

The sampling process to obtain a PCM file introduces errors, because the shape of the original signal has continuous values, and after sampling it gets quantized. With a PCM format it is only possible to obtain a (hopefully) good approximation of the original input signal.

The introduced errors are due to:

- frequency cut at Nyquist$^{(*)}$ frequency.
- approximation of the value of the sample.
- jitter caused by little imprecisions in the clock pulse used to perform the sampling.

Nyquist frequency

Because of the Nyquist-Shannon sampling theorem, the highest representable frequency without aliasing[1] is half of the sampling frequency. Therefore the signal to be sampled should filtered with an antialiasing filter [2] in order to kill frequency above Nyquist frequency. In practice those filters have a transition band of some kilohertz, so they must be applied starting from frequency below the critical ones.

An audio CD will therefore hardly have any energy in frequencies near the critical one (22050 Hz in this case).

[1] Aliasing causes high frequency signals to be reflected onto the lower frequencies, disturbing them

[2] Analogue! The filter is applied before sampling.

Sample approximation

The sample is then represented with a machine number, typically an integer. This introduces approximation errors when the machine number has a small set of values. A size of 16 bits is good, but a trained human ear can perceive the difference. Over 24 bits the human ear cannot distinguish differences. Adding more bits is useless since thermal noise[3] doesn't allow to reach higher precision anyway.

[3] Thermal noise is present in any real system above 0°K, and is caused by fluctuations in the charge carriers inside conductors.

Jitter

Jitter is caused by little imprecisions in the clock[4] used to perform the sampling. Samples should be gathered at regular intervals. On the other hand, the pulse generator used to gather the samples is not perfect, therefore the samples are not really gathered at regular intervals. The digital representation however considers the samples as if they were gathered at perfectly equal intervals, and that obviously introduces errors in the signal.

[4] The periodic pulse

Size of the stream

The size of uncompressed formats depends on the quality (size) of the samples, sampling frequency, and number of channels. To be fairly transparent to the human ear, those parameters need to be quite high. An audio CD has a bitrate of 1411.2 kbit/s[5].

Therefore a PCM format at acceptable quality occupies a lot of space.

[5] 44100 16-bit samples every second for 2 channels = 44100·16·2 =1411200 bit.

Many compressed formats were invented in the past in order to represent digital audio in a compact way. Compressed formats take advantage of the intrinsic redundancy in (digital) audio and psychoacoustic models in order to achieve a good compressing ratio without losing quality (for lossless formats) or losing quality in a graceful way (for lossy formats).

Usually compression is performed in segments, small time windows that are processed and compressed in independent packets (often called frames). Generally compression windows overlap, in order not to create perceptible noises at frame boundaries.

Lossless formats

Lossless formats are basically data compressors, just like the various zip compression formats, since after decompression the original is completely recovered. Zip-like algorithms are too generic to take advantage of the intrinsic redundancy of audio files.

For example, in a stereo audio file it is highly probable that the two channels are very similar. It is possible then to represent more compactly, for example decomposing the signal in sum and difference of the two channels.

Lossless encoding can be a slow process, especially if a good compression ratio is desired. Decoding is very fast, instead; the usefulness of a lossless format with a slow decoding would be very limited

On average a lossless format can compress around 40% - 50%[6], with a big spread, determined by the complexity of the signal to be compressed.

[6] The percentage indicates the amount of information “removed”. From now on this convention will be adopted.

Lossy formats

Lossy formats are somehow similar to lossless formats, that is, they try to take advantage of the intrinsic redundancy of audio files. On the other side lossy formats have a fundamental difference: in order to achieve even greater compression ratios, (non-redundated) information is discarded from the source.

Exploiting psychoacoustic models, less perceptible information is removed, reducing the complexity of the signal and enabling bigger compression ratios.

Very often lossy compression alterates the original signal in a perceptible way, especially when a medium or low quality is chosen (which happens often). The easiest simplification is frequency cutting: the signal is filtered with a low-pass filter, killing the higher frequencies, generally less perceived, especially by adults. Depending on the quality and algorithm, compression artifacts may be introduced, like metallic sound and reverber, and pre-echo.

Compressing at a decent quality, it is possible to achieve a compression ratio 90% - 95%.

Many lossy formats have predefined frame sizes. In the past, those formats allowed only for constant bitrate (CBR), using frames of the same size. Afterwards, encoders and decoders evolved and support a form of variable bitrate (VBR) in which the size of every packet is chosen according to a quality parameter. This form of VBR allows only for fixed packet sizes (and formats), corresponding to the fixed bitrates allowed by the format [7].

There are some formats (like Vorbis) in which the size of the frame is completely free and is only determined by the target quality and the complexity of the fragment to be encoded.

[7] For example, MP3 format supports only the following bitrates: 32, 40, 48, 56, 64, 80, 96, 112, 128, 144, 160, 192, 224, 256 e 320 kbit/s, so the size of packets can be chosen only among 15 different options.

The methods used for audio compression usually belong to two categories: methods that work in the time domain and those that work in the frequency domain.

In the time domain

In the time domain linear predictive codes are use; the encoder uses a model of the source to predict the signal. The format thus contains some parameters for the model, in order to perform the linear prediction, and additional data to compensate for prediction errors.

This formats are generally used to compress human voice; Speex[8], for example, uses linear prediction codes.

[8] Speex is an audio format optimized for human voice compression and it is developed by the Xiph.org foundation, the same that backs Vorbis development.

In the frequency domain

All known formats that work in the frequency domain use the Modified Discrete Cosine Transform (MDCT) to perform the transformation to the frequency domain from the time domain. In this way it is possible to perform (extreme) simplifications and approximations without changing the timbre of the sound.

Frequency cutting is one of the most frequent simplifications, high frequencies are reduced and/or killed, because generally the human ear is not very sensible to them.

Afterwards, the shape of the spectrum is represented in some way, performing codec-, bitrate- and quality-specific simplifications and approximations.

History and diffusion of Vorbis

The development of Vorbis was started by Christopher Montgomery in 1998, after a letter from Fraunhofer Gesellschaft, in which they informed that they intended to charge royalties for licenses of the patents covering the MP3 format, which they invented and patented.

Afterwards other developers joined, and the Xiph.org foundation was created, with the goal of developing and spreading free multimedia formats, like Vorbis.

Usually Vorbis streams are transported in the OGG container format; for this reason the format is often (and incorrectly) called OGG/Vorbis or even just OGG.

An alternative for the OGG container is the RTP transport protocol, typically used for internet streaming.

The reference implementation for Vorbis has always been available from the website of the Xiph.org foundation under a BSD license, a (liberal) open source license.

Successively, alternative and improved versions of the encoder and the decoder appeared. The most widespread alternative versions are: Tremor, aoTuV, and FFmpeg

Tremor

Tremor is an adaptation of the reference implementation that only uses integer arithmetic. This allows decoding of Vorbis on embedded devices, which usually are not very powerful and often lack an FPU.

It is often used on regular processors too, because generally integer arithmetic is faster than floating point.

aoTuV

aoTuV[9] is a modified version of the reference encoder which allows for more compression and more perceived quality. In order to do that, the encoder tries to locate the most “difficult” segments, and temporarily raises the quality. That allows to reduce most of the most common sound artifacts in Vorbis, like pre-echo. Lancer is a modified version of aoTuV optimized to take advantage of multimedia instructions found in modern processors.

[9] aoTuV is an acronym that means “Aoyumi’s Tuned Vorbis”, from the name of the author, Aoyumi (蒼弓).

FFmpeg

FFmpeg is a very versatile audio and video codec library. It supports most audio and video formats, and obviously it supports Vorbis. FFmpeg's encoder is not based on the reference encoder. FFmpeg's is faster.

Since Vorbis is a free and open format, and it is not covered by patents, it is the reference format in the open-source and open-content world. Every audio file on Wikipedia is in Vorbis, and every Linux distribution that includes a media player is capable of decoding Vorbis (MP3 format is not always supported out of the box).

Since there are no royalties to be paid to distribute a Vorbis player, videogames developers often use it as audio format for their products [10].

[10] Some of the titles that use Vorbis for sound effects end/or soundtrack are: “18 Wheels of Steel”, “Unreal Tournament”, “Mafia: The City of Lost Heaven”, “Grand Theft Auto: San Andreas”, “Halo” and “World of Warcraft”. An updated list of videogames that use Vorbis can be fount at: `http://wiki.xiph.org/index.php/Games_that_use_Vorbis`.

The name Vorbis derives from the name of the homonymous character from Terry Pratchett's book "Small Gods", part of the popular Discworld series, which Christopher Montgomery, the creator of Vorbis, likes very much.

Figure: Vorbis

Characteristics of the format

Vorbis is an audio format based on the modified discrete cosine transform (MDCT). The format is designed to allow the addition of a hybrid wavelet-based filter, for Vorbis II, in order to get a better response in signals with localized events.

The format is designed to be encoded by a relatively complex encoder and to be decoded by an (ideally) simple and fast decoder.

Vorbis is computationally more complex to decode than MP3, and it requires more memory to decode because it does not use a static probability model.

Vorbis supports gapless playback.

Vorbis is a variable bitrate format; packet size is not fixed and depends on the quality parameter. The quality parameter ranges from -1 (-2 in some implementations, like aoTuV) to 10. The parameter can be specified in floating point.

The encoder supports different encodings modes:

- quality based: the encoder uses the quality parameter to determine the upper bound of the compression error.
- average bitrate: the encoder tries to keep the specified average bitrate.
- variable bitrate with bandwidth limited: the encoder compresses respecting the upper (and lower) bounds specified.

The best quality/size ratio is obtained with the first mode, followed by the second and the third.

Quality	Xiph.org	aoTuV
-2	-	32 kbit/s
-1	45 kbit/s	48 kbit/s
0	64 kbit/s	
1	80 kbit/s	
2	96 kbit/s	
3	112 kbit/s	
4	128 kbit/s	
5	160 kbit/s	
6	192 kbit/s	
7	224 kbit/s	
8	256 kbit/s	
9	320 kbit/s	
10	400 kbit/s	

Table: Nominal bitrates for Vorbis with stereo input at 44.1kHz

Nominal bitrates were specified in order to estimate the size of a file compressed with Vorbis, and the encoder usually reached those. Subsequently, compression efficiency has improved; now the same file compressed at the same quality is smaller. The quality parameter indicates only the quality and is not bound to the bitrate.

Free format

Vorbis is a free format. Its specifications are free and open, and it is not covered by patents. The Xiph.org foundation releases the reference implementation of the codec under a BSD license.

The Xiph.org foundation reserves the right to certify conforming implementations.

Vorbis audio quality is very high even at low bitrates. Over 160 kbit/s almost every audio formats reach transparency, using the best available encoders. At lower bitrates the differences between the codecs (and the encoders) are accentuated. Several listen tests confirmed the good quality of Vorbis, especially at lower bitrates.

The reference encoder, and all the ones derived from it, are very fast. Their encoding speed is several times faster than LAME[11].

The encoder makes heavy use of floating point arithmetic. At the moment there are no fixed point (integer) arithmetic encoders. The encoder will consequently be very slow on processors lacking an FPU. Those processors are generally used on embedded devices, like media players, that are generally used to play (decode), and not to encode.

Decoding is more computationally intensive than MP3; even using Tremor, more computing power is needed. This means that it is impossible to play Vorbis in real time on some devices that can instead play MP3.

[11] Those valuations are empirical and based on informal tests. Results are consistent, though.

Since it can use UDP as a container, Vorbis is easily streamable. But that's not the only characteristic that makes it useful for streaming.

Because of the format of the packets, Vorbis has a unique characteristic. It is possible to perform "peeling" of Vorbis packets, in order to dynamically reduce the bitrate without re-encoding.

Current implementation are simple (and crude): they just drop the terminal part of the packet[12] until the desired size is obtained. The quality of the result is at the moment lower than re-encoding. Better implementation are being developed, but surely without haste.

Usefulness of peeling for streaming is enormous: it would not be necessary to keep the same file compressed at different bitrates (moreover, usually it's only 2 or 3 different bitrates). One file would be enough, from which a stream of the desired bitrate is extracted. This allows for arbitrary bitrate streams, according to the needs of the client.

[12] The format specifies that an incomplete packet must be decodable.

In the headers of every Vorbis stream there are the coding tables used to represent coefficients used in the process. Those tabled can be quite big and decoders speed- or space-optimized decoders can need even 30KB of additional memory to decode. Such a big requirement is not irrelevant in an embedded device, even though technological evolution is quickly overcoming that problem.

Coding tables are thus created ad-hoc for every stream and are placed at the beginning of the file. This is not very nice for streaming applications, as every connecting client needs to receive the headers in order to be able to decode the rest of the stream. Other formats that have fixed decoding tables, obviously, don't have this problem.

To remedy to that problem, it is possible to implement a server that sends headers at the beginning of each session, or it is possible to just send the tables at regular intervals, so that the client will eventually receive it.

Noise normalization

In order to compensate for the loss of energy at some frequencies due to quantization, the encoder adds noise to the frequency bands that suffered the biggest losses. Since a big loss due to quantization generally implies a low quality, noise normalization only happens when the quality parameter is below 4. This is one of the reasons why compression artifacts in Vorbis sound like rustling, contrary to MP3, which exhibits metallic sounding compression artifacts.

In this section both the reference encoder and the stream format will be analyzed. Because of the structure of the format, it is difficult to produce an encoder too different from the reference.

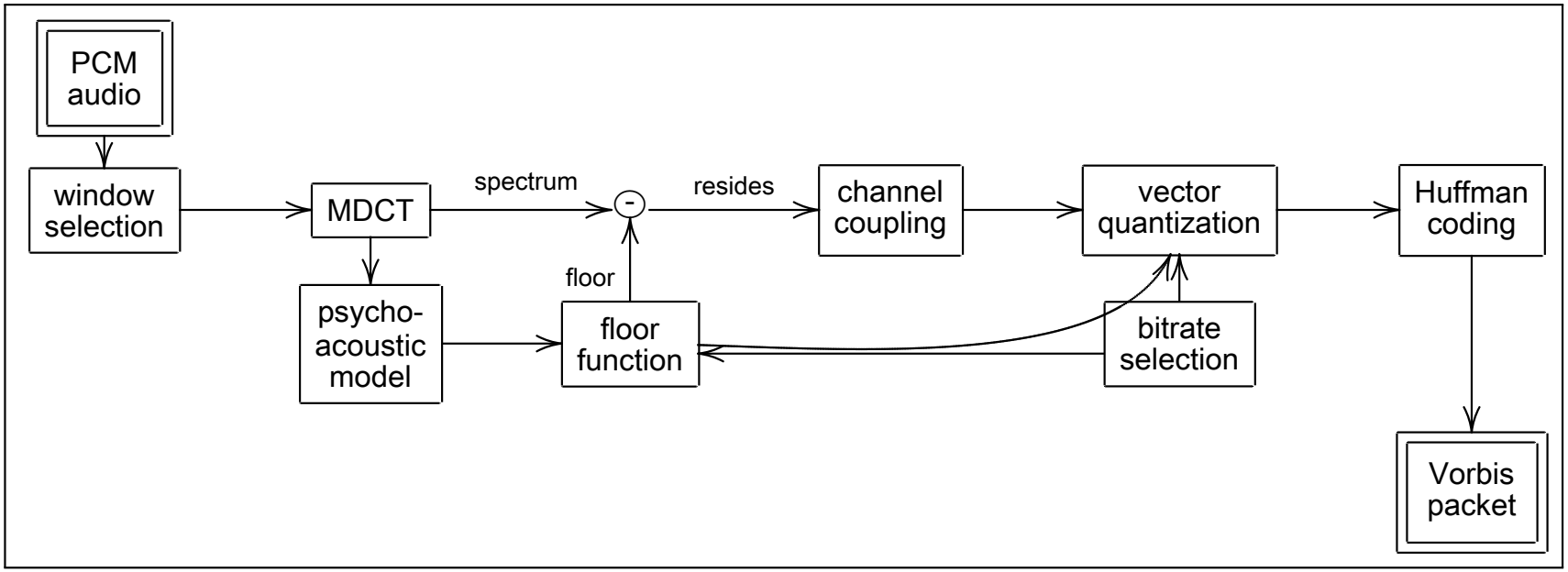


Figure: Diagram of the compression algorithm of Vorbis.

Things that will be covered:

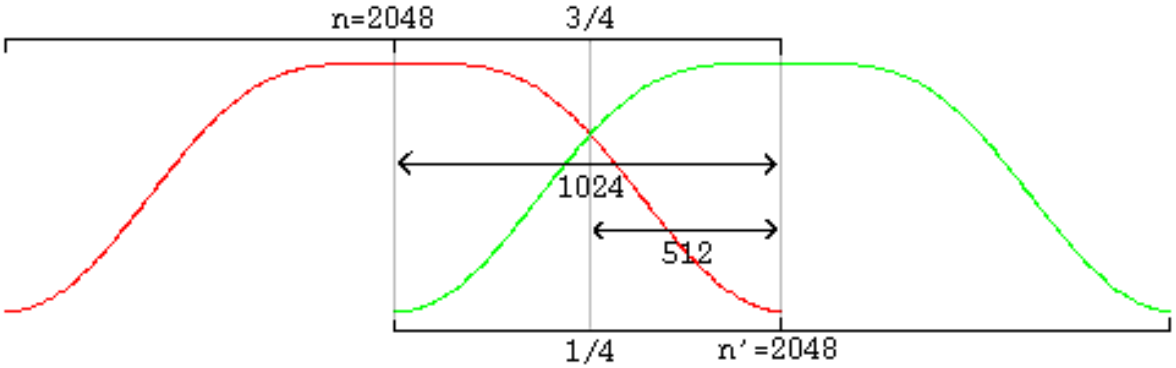
1. Window selection: the right window type is selected and the window is used.
2. Psychoacoustic model: it determines what frequencies must be encoded faithfully in order to sound good to the human ear.
3. Floor function: a low-resolution representation of the spectrum is generated.
4. Generation of residues: the floor is subtracted from the spectrum, residues will be encoded separately.
5. Channel coupling: takes advantage of the redundancy between channels to compress them better and simplify the signal.
6. Vector quantization: the residues Na the floor are compressed with vector quantization, using a code table generated ad-hoc
7. Huffman encoding: the whole thing is compressed using Huffman encoding in order to squeeze it even more.

Every window can have one of two different sizes, specified in the header of the s-tram. Admissible sizes (in samples) are all the power of 2 between 64 and 8192.

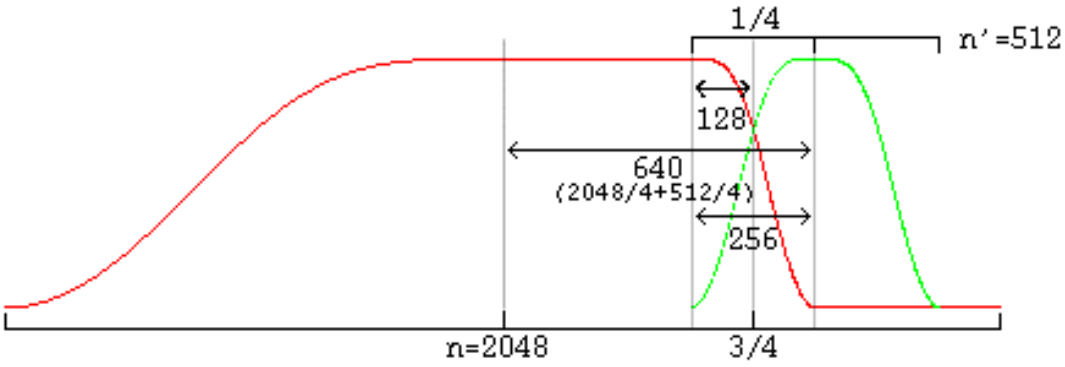
The window function is (n indicates the size in samples):

$$y = \sin\left(0.5 \cdot \pi \sin^2\left(\frac{x + 0.5}{n} \cdot \pi\right)\right)$$

Windows are overlapped so that the point at 3/4 of the first coincides with the point at 1/4 of the second, effectively overlapping the second half of the first with the first half of the second, if windows have the same size. If windows have different sizes, the shape changes a bit. Figure 4 shows both cases.



(a) Equal-sized windows



(b) Differently sized windows

Figure: Examples of overlapping windows.

Psychoacoustic model is based on the absolute threshold of hearing. That model takes advantage of the poor sensibility of the human ear to some frequencies. Figure 5 shows a graph of the absolute threshold of hearing.

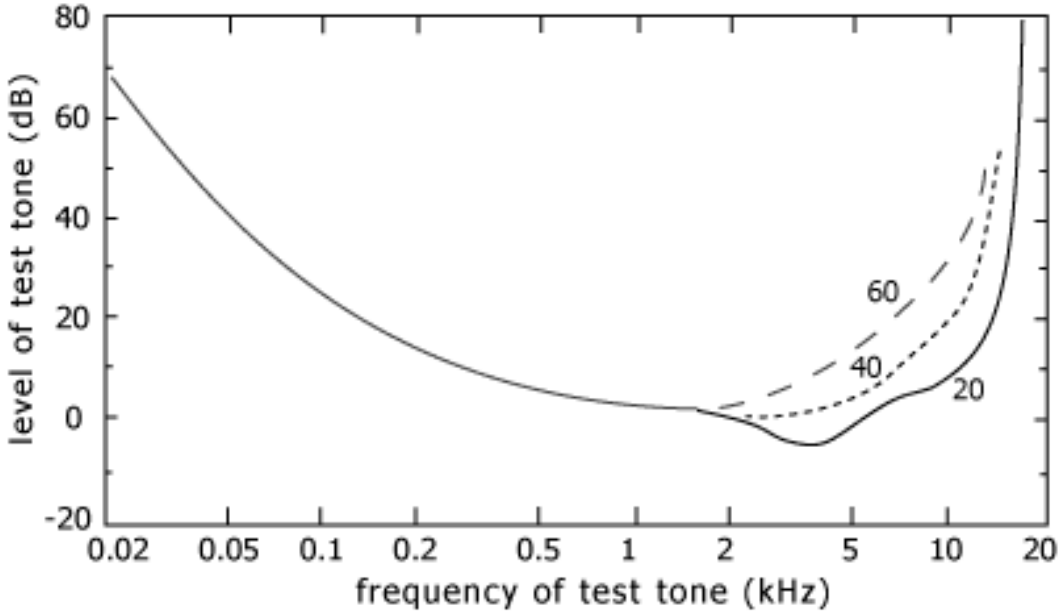


Figure: Graph of the absolute threshold of hearing in function of the frequency of the signal, for different age groups.

The psychoacoustic model will then indicate which frequencies can be sacrificed during compression without a big audible loss of quality, and which should be encoded more faithfully.

Moreover, the phenomenon of masking allows to kill low energy frequencies near other higher-energy frequencies. To model that masking, empirically improved Ehmer curves are used.

The floor function is used to obtain a low-resolution approximation of the spectrum of the window. The floor thus obtained will be then subtracted from the spectrum, obtaining the residues.

Vorbis supports 2 different types of floor functions: type 0 and type 1.

Type 0

Type 0 floors are not any longer part of any encoder, but are still part of the specification, and a decoder needs to be able to decode them in order to be compliant.

The model for type 0 floors is the Line Spectral Pairs model (LSP). That model is used to represent the frequency response in Bark scale of an LSP filter whose parameters are specified in the rest of the packet.

Type 1

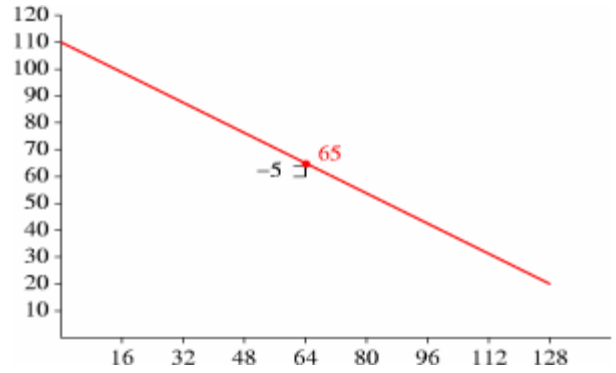
Type 1 floors are the only ones currently used, because they are easier to calculate and present more stability between windows.

The model approximates the spectrum with segments, using the Bresenham algorithm for rasterization of lines.

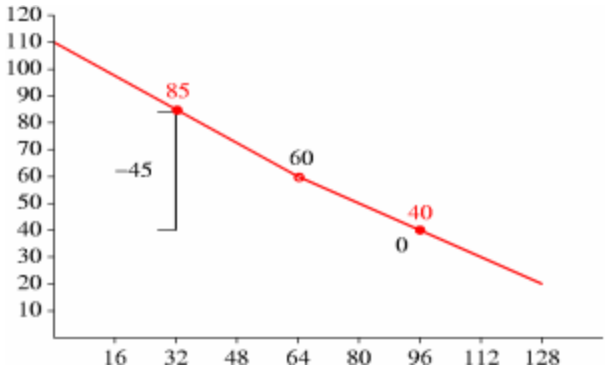
The algorithm approximates the spectrum as a segmented line on a plane, with frequency in a linear scale on the x axis, and intensity in logarithmic scale (in dB) on the y axis, and it's quite simple:

1. An initial line is drawn, both ends are specified.
2. The middle point is taken and the difference between it and the actual value of the spectrum is encoded.
3. The line is subdivided in two segments: the middle point is positioned according to the spectrum.
4. The procedure is reiterated from step 2 on both the segments until the desired resolution is obtained.

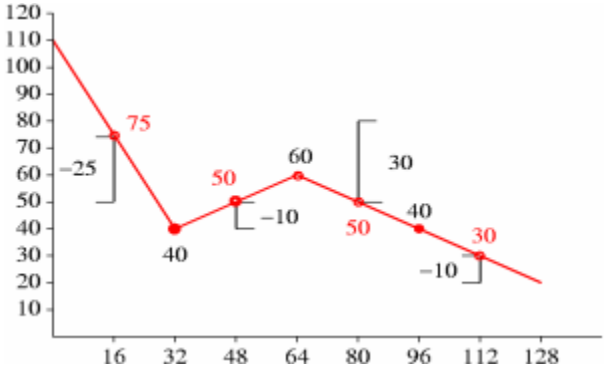
Floor function



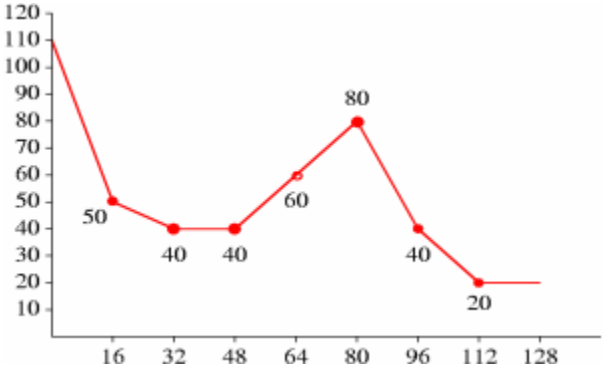
(a) Starting line



(b) After first approximation



(c) After second approximation



(d) And result

Figure: Steps to compute a type 1 floor.

Residues can be encoded in three different ways. The three systems to encode residues differ only marginally. Values of the residues are partitioned in blocks, blocks are then classified and finally classifications and the blocks themselves are encoded.

Encoding of the residues

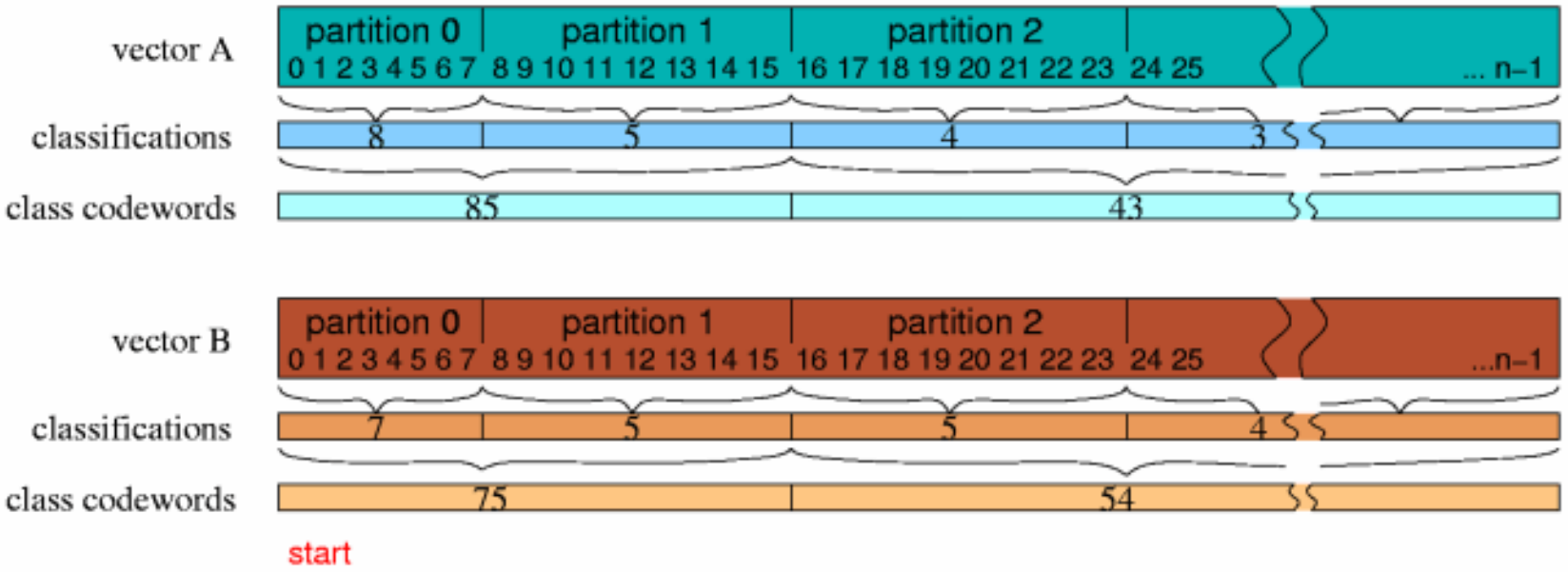


Figure: Scheme of classification of residues. In this example we want to classify 2 vectors, the size of the partition is 8, there are 10 possible classifications and there are 2 classifications per codeword.

Encoding of the residues

1. Every vector is partitioned in equal-sized blocks.
2. Every block of every vector has a classification number that specifies what configuration from the quantization codebook to use. It can be thought that classification numbers form a vector on their own, as indicated in figure 7. Just like vectors of residues, vectors of classifications are divided in blocks, in order to improve compression efficiency.
3. The values in a residues vector can be codified in different ways:
   - single pass
   - multiple passes, representing the vector as a sum of more passes on the vector, using different codebooks for vector quantization.

   The value for the class of every partition does not change at each pass, so the codeword for classification is encoded only in the first pass.

Type 0 residues are encoded alternating the vector quantization encoding according to the dimension of the codebook used to encode a given partition in a specific pass.

Original residues vector: 0 1 2 3 4 5 6 7

size 8 codebook	0	1	2	3	4	5	6	7
size 4 codebook	0	2	4	6	1	3	5	7
size 2 codebook	0	4	1	5	2	6	5	7
size 1 codebook	0	1	2	3	4	5	6	7

Table: Encoding of a type 0 residue.

Table 2 shows an example of encoding of a type 0 residue.

Type 1 residues are encoded like type 0 residues, but without alternating the vector quantization encoding.

Original residues vector: 0 1 2 3 4 5 6 7

size 8 codebook	0	1	2	3	4	5	6	7
size 4 codebook	0	1	2	3	4	5	6	7
size 2 codebook	0	1	2	3	4	5	6	7
size 1 codebook	0	1	2	3	4	5	6	7

Table: Encoding of a type 1 residue.

Table 3 shows an example of encoding of a type 1 residue.

Type 2 residues can be thought as a variant of type 1. Instead of encoding the given vectors as type 1 residues, the vectors are flattened to a single vector, grouping together elements in the same positions, and sorting according to the order of the vectors. Subsequently, the vector is encoded like a very long type 1 vector.

Original residues vector:

0	1	2	3	4	5	6	7
a	b	c	d	e	f	g	h
غ	ل	م	ن	و	ذ	خ	ق

Resulting vector:

0	a	غ	1	b	ل	2	c	م	3	d	ن	4	e	و	5	f	ذ	6	g	خ	7	h	ق

The procedure is then analogous to the type 1 residue.

Table: Encoding of a type 2 residue.

Channels can be encoded independently, without taking advantage of redundancy between channels. The reference encoder tries to take advantage of channel redundancy when their structure and redundancy is known. Generally the encoding is performed on stereo signals; other dispositions are specified (e.g. for 2.1, 4.1, 5.1, 6.1, 7.1, etc.)in in other cases the encoder encodes the channels separately.

Strategies to eliminate redundancy between channels are:

- Type 2 residues
- Square polar representation

Those mechanisms can be used at the same time, an indeed the reference encoder does it.

The first case is trivial: the residues of the two channels are alternated, so that the corresponding residues are near, and let the Huffman encoding do the rest.
The second case is more complicated.

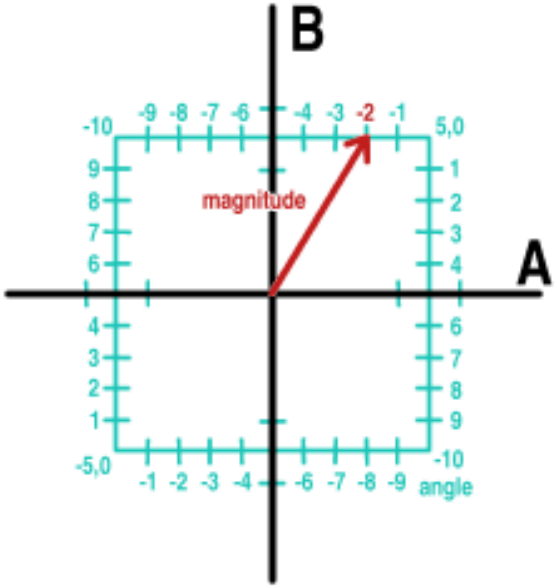


Figure: Square polar representation. A and B are the two coupled channels.

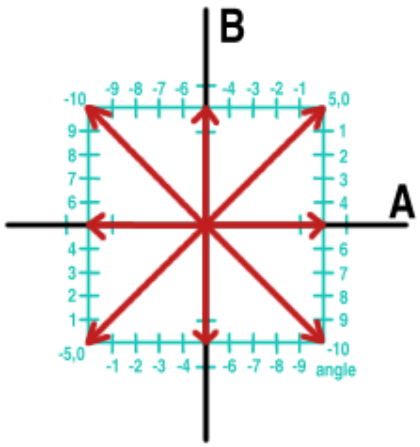
Lossless stereo

Sound is decomposed in intensity and angle (phase). In order to avoid using trigonometric functions during the decoding phase, a square polar representation was adopted, as shown in figure 8. This representation allows to allows for a perfect representation of both channels. This mode is called “lossless stereo”, because the informations regarding phase are completely preserved (but in less space).

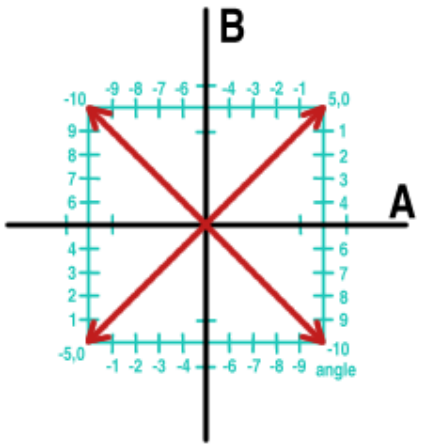
Phase stereo

Since the human ear does not have a good perception of phase for signals well above 4kHz, it is possible to simplify the square polar representation by restricting the choice of possible angles. For example, restricting the choice of angles between 4 possible angles, there are 8 possible permutations, since the intensity can be negative. Figure 9(a) shows this configuration, in which signals can be in phase or out of phase by 90° or 180°.

Further restricting the number of choices for the angle, it is possible to reach the configuration shown in figure 9(b), in which channels can be in phase or out of phase by 180°.



(a) Square polar representation with 4 possible choices for angles.



(b) Square polar representation with 2 possible choices for angles.

Figure: Signal simplifications through square polar representation.

Point stereo

Finally, the last step is to completely ignore angles and to encode only intensity. This simplification can be perceived strongly on low frequencies, especially using earphones (since they provide a good stereo separation).

Interestingly, the reference encoder (and derived encoders) only used the “phase stereo” method until version rc2. Since version 1.0 it uses only a mixture of lossless and point stereo.

Vector quantization is a very powerful data compression tool.

Vectors to be represented are divided in groups of approximately the same size. Every group is represented by the centroid. Multidimensional vectors are then projected in a smaller discrete subspace. Since data is represented by the index of the nearest centroid, more frequent data will have a smaller error while less frequent data will have a bigger error.

Transformation between spaces is usually performed with projection, or, like in Vorbis, using codebooks.

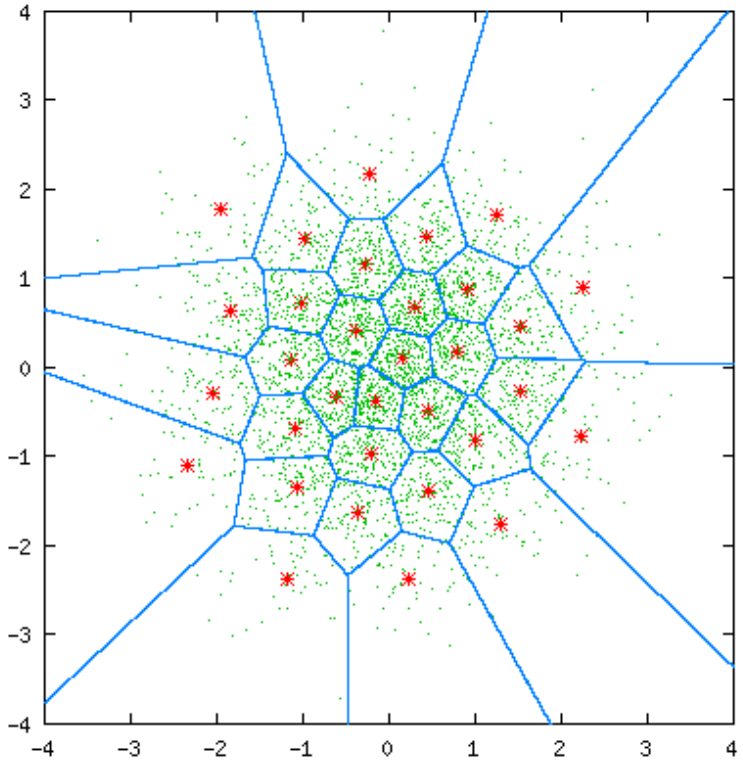


Figure: Example of Vector Quantization

Huffman encoding uses a specific method to choose the representation of every symbol, resulting in an prefix-code

Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code[13] that expresses the most common source symbols using shorter strings of bits than are used for less common source symbols. Huffman was able to design the most efficient compression method of this type: no other mapping of individual source symbols to unique strings of bits will produce a smaller average output size when the actual symbol frequencies agree with those used to create the code.

[13] sometimes called ”prefix-free codes”, that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol

The algorithm is simple:

1. Start with as many trees as there are symbols.
2. Find the two trees with the smaller size (frequency).
3. Combine the two trees in one single new tree, with one of the old trees as left branch and the other as right branch.
4. Repeat from point 2 until only one tree is left.

Huffman encoding

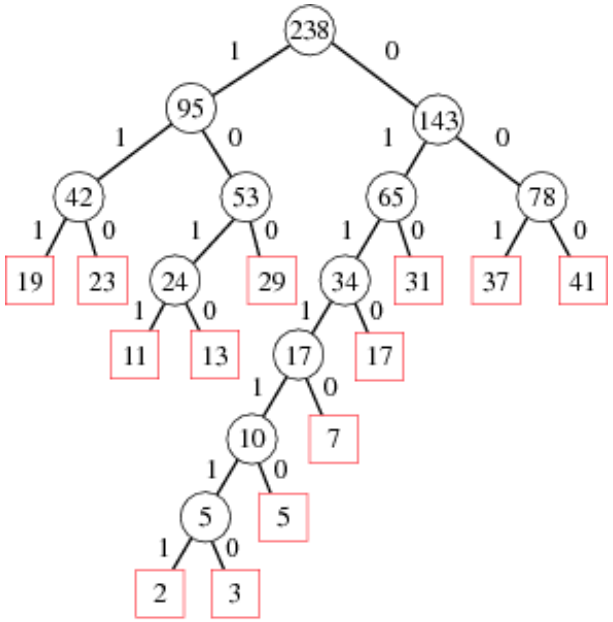


Figure: Example of Huffman tree. The numbers in the squares indicate the frequency of the symbol. The numbers in the circles indicate the total frequency of that subtree.

Deciding what bits encodes the left and right branches is enough. The path to reach every symbol indicates the bit sequence that encodes that symbol.

Listening tests

In order to directly verify how well Vorbis performed, a small and informal listening test was performed. The test was unluckily performed with only 4 subjects.

The test consisted in grading some pieces compressed in different formats and at different qualities. The pieces were first encoded and then reconverted to wav, in order to hide their identity.

Every test subject performed the test by itself, using different earphones or speakers and in different environments. The use of any analysis tool was forbidden, in order to avoid (even unintentional) bias.

Grading

The grade, from 0 to 10, indicated the perceived quality of the piece.

0 means that the piece is not even recognisable.

10 means that the piece is perceived to be the original, or completely equivalent.

Pieces

This are the pieces used (in brackets the type):

- “Innuendo” by Queen
  (rock)
- “Living On My Own” by Freddie Mercury
  (rock/dance)
- Capriccio n. 24 by Paganini
  (classical: violin solo)
- “In the hall of the mountain king” by Grieg
  (classical with choir)
- First movement of the 5th Brandenburg Concert by J. S. Bach, BWV 1050
  (classical: orchestra and harpsichord solo)

Formats and quality

Every piece was compressed from the original (CD-Audio) into the following formats:

- Uncompressed WAV.
- AAC (using FAAC[14]) at following qualities[15]: 350, 200, 100, 40, 10.
- MP3 (using LAME[16]) at the following qualities: VBR preset “medium”, ABR 192 kbit/s, ABR 128 kbit/s, ABR 64 kbit/s, ABR 32 kbit/s.
- Vorbis (using aoTuV beta 5) at the following qualities: 7, 3, 0, -2.

Note that the uncompressed original was present, but it was not indicated. The test was thus a blind test.

The resulting files have the bitrates[17] indicated in table 5.

[14] FAAC is considered at the moment the worst AAC encoder, even by its own authors.
[15] Quality in FAAC is expressed with a number between 10 (minimum) and 500 (maximum).
[16] LAME is considered the best MP3 encoder.
[17] Bitrates were calculated from the final size of the file after compression.

actual bitrates	Innuendo	Living On My Own	Capriccio N. 24	In the hall of the mountain king	Brandenburg concert
WAV	1411.20	1411.20	1411.20	1411.20	1411.20
AAC 350	262.56	287.85	251.14	226.12	254.86
AAC 200	212.30	236.14	199.82	178.38	206.31
AAC 100	133.50	147.46	122.91	108.50	130.96
AAC 40	62.77	63.43	62.56	52.08	64.39
AAC 10	33.69	31.56	36.23	32.77	36.88
MP3 VBR	153.87	164.29	159.51	156.46	146.09
MP3 192	185.46	189.54	184.61	180.77	182.91
MP3 128	123.11	123.70	122.42	118.92	121.43
MP3 64	62.60	62.98	62.07	59.24	62.19
MP3 32	31.59	32.77	31.66	29.95	31.56
Vorbis 7	213.37	249.81	202.54	191.39	219.51
Vorbis 3	110.37	127.16	109.19	105.46	115.45
Vorbis 0	62.35	73.50	59.97	57.50	60.60
Vorbis -2	31.84	37.57	28.93	32.12	33.26

Table: Bitrates of the compressed pieces (in kbit/s).

Tables 6, 7, 8 and 9 contain the grades of the single test subjects, while table 10 contains the mean grades.

The mean was calculated with a simple arithmetic mean.

First subject	Innuendo	Living On My Own	Capriccio N. 24	In the hall of the mountain king	Brandenburg concert
WAV	9	8	8	9	10
AAC 350	6	9	9	10	8
AAC 200	9	10	9	9	9
AAC 100	9	9	6	9	7
AAC 40	4	5	2	10	5
AAC 10	0	1	0	4	1
MP3 VBR	9	9	9	9	7
MP3 192	10	8	8	7	6
MP3 128	10	8	10	8	7
MP3 64	10	7	8	8	9
MP3 32	8	5	8	7	7
Vorbis 7	8	8	9	9	9
Vorbis 3	6	8	6	8	9
Vorbis 0	8	8	10	8	9
Vorbis -2	6	4	10	8	8

Table: Results of listening test for the first subject.

Second subject	Innuendo	Living On My Own	Capriccio N. 24	In the hall of the mountain king	Brandenburg concert
WAV	10	10	10	9	10
AAC 350	9	9	9	10	9
AAC 200	9	9	9	9	9
AAC 100	9	9	9	9	5
AAC 40	3	3	3	3	1
AAC 10	1	1	0	0	0
MP3 VBR	7	8	9	9	9
MP3 192	9	9	7	9	9
MP3 128	9	9	8	8	4
MP3 64	8	5	9	5	6
MP3 32	5	2	1	1	3
Vorbis 7	9	9	9	9	8
Vorbis 3	9	9	8	9	9
Vorbis 0	8	8	9	6	7
Vorbis -2	7	6	3	6	8

Table: Results of listening test for the second subject.

Third subject	Innuendo	Living On My Own	Capriccio N. 24	In the hall of the mountain king	Brandenburg concert
WAV	10	9	10	10	9
AAC 350	9	9	9	10	9
AAC 200	9	10	9	9	9
AAC 100	9	9	9	9	9
AAC 40	5	4	6	4	7
AAC 10	0	0	0	0	1
MP3 VBR	9	9	10	9	10
MP3 192	9	9	9	10	9
MP3 128	9	9	9	9	9
MP3 64	8	6	9	5	8
MP3 32	2	5	8	1	2
Vorbis 7	9	9	10	9	10
Vorbis 3	9	9	9	9	9
Vorbis 0	9	9	9	9	10
Vorbis -2	8	7	9	7	8

Table: Results of listening test for the third subject.

Fourth subject	Innuendo	Living On My Own	Capriccio N. 24	In the hall of the mountain king	Brandenburg concert
WAV	9	8	9	9	9
AAC 350	9	9	8	9	6
AAC 200	10	9	8	5	8
AAC 100	5	8	7	8	8
AAC 40	8	5	5	4	7
AAC 10	0	0	1	2	0
MP3 VBR	9	8	7	10	8
MP3 192	8	8	7	9	8
MP3 128	9	9	7	6	9
MP3 64	9	8	7	8	8
MP3 32	5	5	1	6	7
Vorbis 7	9	9	9	9	7
Vorbis 3	8	8	5	9	8
Vorbis 0	9	8	5	7	9
Vorbis -2	9	7	5	8	7

Table: Results of listening test for the fourth subject.

Average grades	Innuendo	Living On My Own	Capriccio N. 24	In the hall of the mountain king	Brandenburg concert
WAV	9.5	8.75	9.25	9.25	9.5
AAC 350	7.5	9	8.75	9.75	8.75
AAC 200	9.25	9.5	8.75	8	8.75
AAC 100	8	8.75	7.75	8.75	7.25
AAC 40	5	4.5	4	5.25	5
AAC 10	0.25	0.5	0.25	1.5	0.5
MP3 VBR	8.5	8.5	8.75	9.25	8.5
MP3 192	9	8.5	7.75	8.75	8
MP3 128	9.25	8.75	8.5	7.75	7.25
MP3 64	8.75	6.5	8.25	6.5	7.75
MP3 32	5	4.25	4.5	3.75	4.75
Vorbis 7	8.75	8.75	9.25	9	8.5
Vorbis 3	8	8.5	7	8.75	8.75
Vorbis 0	8.5	8.25	8.25	7.5	8.25
Vorbis -2	7.5	6	6.75	7.25	7.75

Table: Average grades of the four test subjects.

It can be noted that above 128 kbit/s every codec is transparent.

It's not surprising that the original received high grades. Sometimes the original didn't get the highest grade, while some subjects always recognized the original.

AAC – FAAC

The performance of FAAC is very poor, especially at lower bitrates. At bitrates above 128 kbit/s it reaches transparency, like the other codecs.

There are no other free implementations of AAC encoders, and FAAC has in general very poor performance. It must be said that the encoder is still under development and the developers themselves admit that it is not yet ready for final users.

MP3 – LAME

The performance of LAME is very good. After all, LAME is recognized as the best MP3 encoders, perhaps second only to the one made by the Fraunhofer Gesellschaft, and surely it is the best free encoder.

LAME manages to obtain good results with few artifacts. To do that it applies rather aggressive low-pass filters. Those frequency cuts are perceived mostly at lower bitrates; at those bitrates the faintest sounds simply disappear and in general the overall sound is unclear and pasted, and the single instruments can barely be distinguished.

Vorbis – aoTuV

Vorbis performance is very good, even at low bitrates. It even got higher grades in pieces compressed at quality 0 than quality 3.

Even though quality -2 amplifies high frequency noises and creates strange out-of-phase effects in some stereo fragments, it received high grades, considering the bitrate.

The sound of Vorbis is characterized by a slim sound, without evident frequency cuts; on the contrary, at the lowest bitrates higher bitrates are amplified even too much.

Final considerations

Vorbis is a very good audio codec, and it already outdoes MP3 with respect to perceived audio quality.

It can still be improved. The developers continue to improve the quality of the encoder, and the wavelet filters are still missing.

In conclusion, Vorbis is already a very good format and it has very big room for improvement.

References

Sites:

Wikipedia
(*www.wikipedia.org*)

General informations and links to other resources

Xiph.org
(*www.xiph.org*)

Vorbis official site, with informations and complete specifications of Vorbis

HydrogenAudio
(*www.hydrogenaudio.org*)

Detailed informations regarding Vorbis, and link to other resources and other listening tests

Masking Patterns of Tones
(*http://www.zainea.com/masking2.htm*)

Ehmer masking curves

Ogg Vorbis: Subjective assessment of sound quality at very low bit rates
(*http://www.cesnet.cz/doc/techzpravy/2006/vorbis/*)

Informations and figures

References

Publications:

Appunti di Teoria dell'Informazione versione 2.3, Ottobre 2001 by Pietro Piram e Francesco Romani

A method for the construction of minimum-redundancy codes Proceedings of the I.R.E., Settembre 1952, pagg. 1098-1102 by D. A. Huffman

Digital Signal Processing - A Computer Science Perspective (ISBN 0-471-29546-9) by Jonathan Stein

Questions?

License