EXPERIMENTAL AUDIO CODER: TESTING IMPLEMENTATION

F. Rund, J. Nováček

Department of Radio Engineering, Faculty of Electrical Engineering,
Czech Technical University in Prague

Abstract

This article results from the last year article at the conference, which designed structure of an experimental audio encoder. The designed structure was implemented by Matlab functions, each function represents one system block. Consequently was made testing implementation of a simple perceptual audio coder which respects structure that was designed in the article.

1 Introduction

Considering almost the same inner structure of all perceptual audio coders, the universal audio coder was developed in our last year article [1]. In this article is presented simple implementation of the perceptual audio coder to test the structure designed in [1].

Presented simple encoder substitutes psychoacoustic model by a static one which takes into account frequency dependency of the hearing threshold only. Masking effects are not considered by this static psychoacoustic model, which is not usual but demonstrative enough. Coding is performed in the frequency domain that is obtained by the Fast Fourier Transform.

Frequency components are then re-quantized with a different bit-length according to the designed psychoacoustic model. Appropriate decoder was also implemented to prove correct function of the encoder.

The coder was designed as simple as possible to demonstrate fundamentals of perceptual audio coding. As was expected coder's compression rate is very low and perceived quality is poor, because coded audio signal suffers from the origin of artifacts. Modular design of the coder enables further improvement of the particular blocks or whole system.

2 Coder implementation

In our last year article [1] we designed structure of an experimental audio encoder. The article was based on the fact, that the most audio-compression systems deals with the same psychoacoustic phenomena, so their inner structure is similar (see e.g. [2], [3], etc.).

The coder is a block based as shown in Fig. 1, each block is implemented as a Matlab function, as was proposed in [1]. The name of each function consists of the letter `K`, followed by the key-word specifying the system block and ending by the key-word specifying the implemented algorithm. For example `Ktimefreq_fft1` denotes function which realizes the block Time/Frequency Transformation using the `fft` (Fast Fourier Transform) algorithm.

The whole coder was implemented as a Matlab script, which uses functions described below. Because of a coding in the frequency domain it is necessary to use overlap-and-add windowing technique before the signal is transformed from the time to the frequency domain (see e.g. [2]). The data is divided into blocks and transformed into the frequency domain by this procedure. The main part of the coder is a `for`-loop which performs encoding of each block of the encoded audio data. The block of 1,024 samples is read from the input audio file, transformed into frequency domain, quantized along the psychoacoustic model results, and written to the

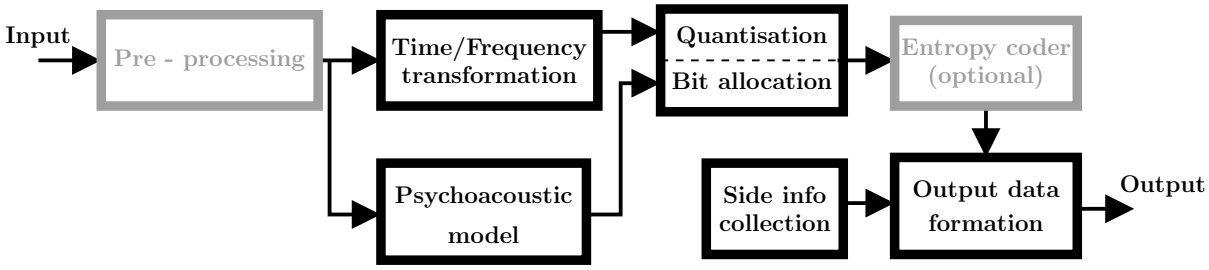

Figure 1: Block diagram of the encoder as was defined in [1]. Grey-colored blocks are bypassed in the implementation.

output (compressed) file. The 50% overlapping of 1024 samples windows is used in the coder's implementation.

Due to the fact that the psychoacoustic model is static (independent of the input data) it is sufficient to compute the model only on the beginning of the coding. It is also sufficient to transfer parameters of the psychoacoustic model, e.g. weighting window, bit-allocation, etc., to the decoder in the header of whole coded file only (not in each block). So, the task is a little bit simplified, but demonstrative enough.

2.1 Preprocessing

Optional pre-processing block as was defined in the article [1] is not used in the implementation therefore is bypassed and in the block diagram in Fig. 1 is shown by a gray box. An input signal is expected to be pre-processed (e.g. normalized) externally.

2.2 Time/Frequency Transformation

Coding is performed in the frequency domain that is obtained by the Fast Fourier Transform. Coded signal is windowed by the Sine Window before the Transform is performed. Importance of a signal windowing is described in detail in [2].

The function is declared, as was proposed in [1], as follows
`function [data_out,data_descr,info_tf,varargout] = Ktimefreq_fft1(data_in,fs)` ,

The name of the function means, that the Fast Fourier Transform was used for a conversion from time do frequency domain. The input data (`data_in`) with sampling frequency `fs` are multiplied by a sine window and the Fourier Transform is computed. The main output of the function is the `data_out` vector, with spectral coefficients. Vector `data_descr` containes description of the data, in this case it is the frequency vector. By the parameter `info_tf` is information about the Window shape transferred to the decoder. The `varagout` parameter is not used in this version.

2.3 Psychoacoustic Model

Presented simple encoder substitutes psychoacoustic model by a static one which takes into account frequency dependency of the hearing threshold only. Instead of the hearing threshold is used the equal-loudness curve [4]. Used equal-loudness curve may be changed by a parameter to adjust coder performance. Masking effects are not considered by this static psychoacoustic model, which is not usual but demonstrative enough as the coder is used for educational purposes not for efficient audio coding.

Function declaration:
`function [thres_f, thres_spl, info_pm, varargout] = Kpsychomod_cu25(data_in)` ,

The name of the function denotes, that the 25 Phones equal-loudness curve is used as a threshold. The outputs of the function are not dependent on input data, so it is not necessary to compute the model for each block of data. The outputs of the function describe output threshold curve. The `thres_f` are frequencies, where the threshold `thres_spl` is defined. Used equal-loudness curves are defined up to 15 kHz, so higher frequency components are erased by the encoder. The parameter `info_pm` is not necessary for the decoder therefore is not stored. The `varagout` parameter is not used in this version.

The threshold used in this simple encoder is shown in Fig. 2

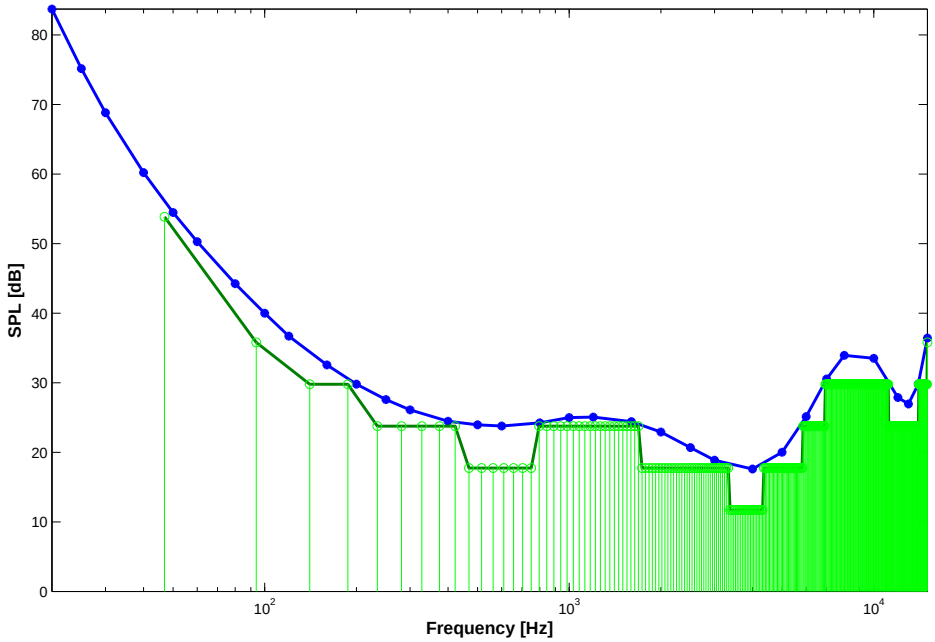

Figure 2: Used 25 Phones threshold curve with a frequency components bits-per-sample reallocation scheme.

2.4 Bit Allocation and Quantizer

By this part of the coder, frequency components are re-quantized with a different bit-length according to the designed psychoacoustic model.

The function is declared as follows:

```
function [data_out, descr_out, info_abk, varargout] = ..
         Kalokqant_kr1(data_in, descr_in, thres_spl, thres_f, varargin) ,
```

It is evident, that the inputs are data from Time/Frequency Transform (`data_in, descr_in`) and Psychoacoustic Model (`thres_spl`, `thres_f`).

Bit Allocation in this case is also static and is derived from the threshold given by the psychoacoustic model. At first, the threshold levels are interpolated at the frequencies, where spectral coefficients of the data are defined (up to the maximal frequency of the threshold – 15 kHz in this case). Then, from the threshold, the number of bits for each spectral coefficient is computed, as is shown in Fig. 2.

In Quantization part of the block, each spectral coefficient (`data_in`) is re-quantized according to the allocated number of bits. No scaling factors are used in this simple version of the coder. The outputs are re-quantized spectral coefficients (`data_out`) along with a corresponding frequency vector (`descr_out`). For the test purposes is transferred information about designed

quantization in `info_abk` variable.

The `varagout`, `varagin` parameters are not used in this version.

2.5 Entropic coder

Entropic coder is not used in this simple example. It can be modeled by using of Matlab standard `save` function, which uses entropic coding for saving in `.mat` files.

2.6 Collecting of Side and Auxiliary Info

Information from all parts of the coder is collected for usage in the decoding process. For our simple coder it is necessary to transfer only information about the Windowing function and Transformation used in the Time/Frequency analysis part and about the number of bits for each spectral coefficient. All this information is constant for this model, so it is not necessary to transfer it into decoder more than in the header of the whole coded file.

Function can be declared as follows:
`function [info, descr_info] = Ksideinfo_simple(varargin)` ,

but, as was noted, for this simple coder is not necessary to transfer any side or auxiliary information within the coded data.

2.7 Output Data Stream Formatting

As it is not necessary to transfer any side information in this simple coder, only quantized spectral coefficients are needed to be transferred.

The function can be declared as follows:
`function [status] = Kformout_simple(info,descr_info,in_data,descr_in,varargin)` ,

but, only the `in_data` vector will be formed into output file block. The other inputs are static (the same for all data blocks) and are transferred only on the beginning of the file. The output of the function is zero, if the coding was succesfull.

3 Coder performance

The coder, using the above defined functions, was developed in Matlab and tested by real sound samples. In the Fig. 3 is shown quantization noise produced by the coder. It is evident, that the noise is shaped along the desired curve. The performance of the coder can be computed as follows. The number of bits which is needed to be transferred without using of compression system is given by a multiplication of a number of samples and the bit-depth. For the 1,024 16 bit samples we thus need transfer 16,384 bits. In the designed coder, the frequency range is reduced only to 15 kHz, which means that we need to transfer only 642 instead of 1024 spectral coefficients. If is used 16-bit quantization, it would give 10,272 bits. But we only use from 7 to 14 bits per coefficient, so it can be 7,682 bits. But, when we use 50% overlapping, it is 15,364 bits. So, our coder has very low performance, about 6%.

The performance can be increased, when we use higher equal-loudness curve, but it also decreases the perceived audio quality. For better performance, better psychoacoustic model (e.g. masking phenomenon) is necessary. Also for Time/Frequency Transform are more suitable transforms, e.g. widely used DCT (Discrete Cosine Transform), or another approach to quantization of spectral coefficients (imaginary parts of coefficients can be quantized differently than

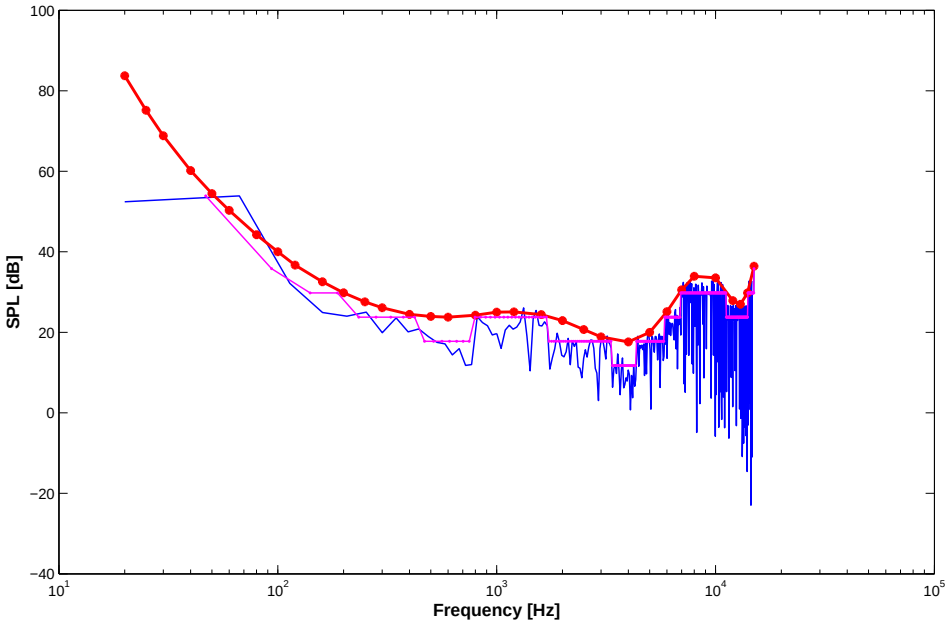

Figure 3: Used threshold curve with a quantization noise produced by the coder.

real parts) can be provided. Also using of scaling factors and other techniques can increase the performance.

4 Decoder

Appropriate decoder was also implemented to prove correct function of the encoder. The block diagram of the decoder is in the Fig. 4. It is apparent, that the function of the decoder is reversal to the coder.

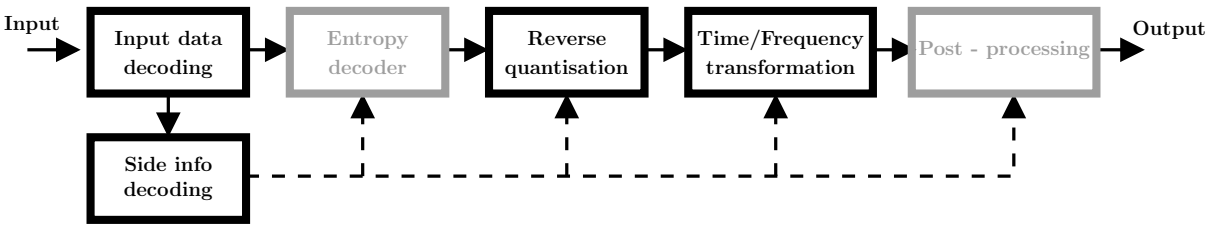

Figure 4: Block diagram of the decoder. Grey-colored blocks are bypassed in the implementation.

The sample of decoded signal in comparison with the original one can be seen in Fig. 6. In the picture is the added noise seen. Of course, the noise is also heard, but the perceived quality is not as poor as we expected.

5 Conclusion

The coder was designed as simple as possible to demonstrate fundamentals of perceptual audio coding principles. As was expected coder's compression rate is very low and perceived quality is not the best, because coded audio signal suffers from the origin of artifacts, namely the added noise. Perceived quality depents on the type of music a lot Modular design of the coder enables

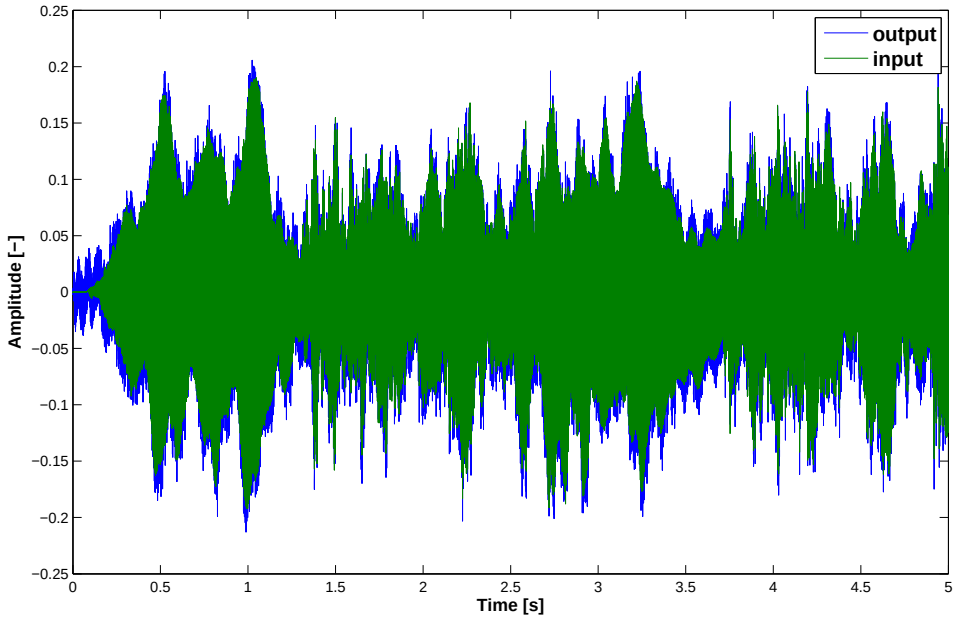

Figure 5: Comparison of original (green) and decoded (blue) signal.

further improvement of the particular blocks or whole system. Some tips how to increase the compression rate was mentioned above.

Acknowledgement

This project was supported by the research program MSM 6840770014
"Research in the Area of Prospective Information and Communication Technologies".

References

[1] Rund, F. - Nováček, J.: Experimental Audio Coder. In Technical Computing Prague 2007 [CD-ROM]. Prague: HUMUSOFT, 2007, ISBN 978-80-7080-658-6. (in Czech).

[2] Bosi, M., Goldberg, R. E.: *Introduction to Digital Audio Coding and Standards*, Kluwer Academic Publishers, USA 2003.

[3] Painter, T., Spanias, A.: *Perceptual Coding of Digital Audio*, In IEEE Proceedings, Vol 88, No. 4, pages 451-513, April 2000.

[4] Zwicker, H., Fastl, H.: *Psychoacoustics*, Springer-Verlag, Berlin, 1990.

František Rund
Department of Radio Engineering, FEE, CTU Prague, Technická 2, 166 27, Prague 6
tel. 22435 2108, e-mail: xrund@fel.cvut.cz

Jan Nováček
Department of Radio Engineering, FEE, CTU Prague, Technická 2, 166 27, Prague 6
tel. 22435-2109, e-mail: novacj1@fel.cvut.cz