Department of Electrical Engineering
McGill University

Experiment 4

Subband Audio Compression

4.1 Purpose

In this experiment, we will implement a very basic audio compression program based on subband decomposition and scalar quantizers.

4.2 Background

Audio signals are represented digitally as a series of samples. Usual values for the sampling frequency and number of bits per sample are e.g. 44.1 kHz and 16 bits, respectively. It is clear that the size of a stereo audio file can grow very rapidly: with the values given above, $2 \times 16 \times 44100 = 1,411,200$ bits are required per second of audio. In order to enable transmission or storage of digital audio, bandwidth or space limitations urge for compression of the audio signal.

The principle of lossy audio compression is to exploit the inherent redundancy in the audio signal as well as the limited perceptual capabilities of human hearing in order to transform the digital audio into a more compact format, without introducing noticeable or annoying artifacts. Clearly, compression ratio can be traded off for quality of the compressed signal.

The very basic audio compression system we will consider in the following relies on the fact that most natural audio signals do not exhibit a flat frequency response. It then makes sense to split the audio signal into frequency subbands and to process these subbands separately. In particular, the higher energy subbands are more important for a faithful reproduction of the audio signal. In order to translate this consideration into a useful design strategy, one uses different scalar quantizers to quantize the different subbands. It makes sense to quantize more finely, or more accurately, the higher energy subbands. The principle of *bit allocation* is to devote more bits (i.e. more quantizer reproduction levels) to more important subbands. If b_i is the number of bits used to represent a quantizer reproduction level in subband i, then compression is achieved as long as the average of b_i over all subbands is lower than the original number of bits used to represent one audio sample.

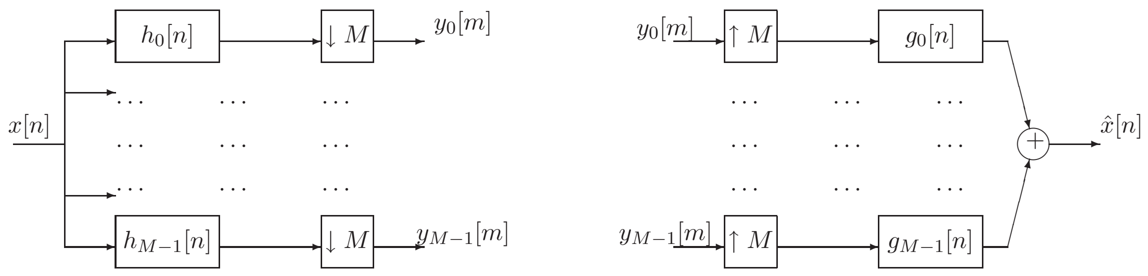

Fig. 4.1 Block diagram of a subband coding system

4.3 Multirate Filter Banks

Fig. 4.1 illustrates a multirate filter bank that uses downsampling and upsampling to split up an input signal into different frequency bands and then reconstruct it. This figure shows *critical* sampling, i.e., in the figure, the sum of the sampling rates of the M channels equals the sampling rate of the input signal.

Filters

Each of the M ideal bandpass filters has a bandwidth of π/M. Their frequency response is given by:

$$H_m(e^{j\omega}) = \begin{cases} 1 & \text{for } \frac{\pi m}{M} < |\omega| < \frac{\pi(m+1)}{M} \\ 0 & \text{otherwise } m = 0, \ldots, M-1 \end{cases} \tag{4.1}$$

Since each filtered signal contains only $1/M$ of the frequency range of the original signal, it can be downsampled by a factor of M. Perfect recovery of the original $x[n]$ can still be achieved from the reduced rate sequence of samples.

Upsampling/Downsampling

The blocks with downward pointing arrows represent the operation of downsampling by a factor of M. Similarly the blocks with upward arrows represent the operation of upsampling.

Upsampling by a factor of M is the process of inserting $M-1$ zeros between successive samples of the input sequence (thus increasing the sampling rate by a factor of M).

Downsampling by a factor of M is the process by which only every M^{th} sample of the input sequence is selected to form the output sequence. This effectively reduces the sampling rate of the input signal by a factor of M. The resulting signal, in the frequency domain, experiences a frequency spreading by a factor of M. Aliasing may occur, but it can be avoided by preceding this operation with a filter that reduces the bandwidth of the input to $\omega_{\max} = \pi/M$. For more details on Decimation/Interpolation see [1, Sections 10.2 to 10.4].

Ideal brick-wall filters are not physically realizable; as a result, frequency components in one branch of the sub-band coder, will leak into other branches. When we come to reconstruct our signal at the source decoder, we will introduce reconstruction errors (even in the absence of quantization errors). It is possible though, to design the analysis filters $h_k[n]$ and the synthesis filters $g_k[n]$ such that the distortion resulting from aliasing and amplitude and phase distortions are minimized.

In order to illustrate the effect of the filter banks, one can express the input-output relationship of the analysis-synthesis cascade either in the time domain or in the z-transform domain. For instance, in the z-domain, we get that

$$\hat{X}(z) \quad = \quad \frac{1}{M}\sum_{i=0}^{M-1}\sum_{k=0}^{M-1} G_i(z)H_i(zW_M^k)X(zW_M^k), \tag{4.2}$$

where W_M is the M-th root of unity. In this equation, two different kinds of terms appear, one term which multiplies $X(z)$, i.e.

$$\frac{1}{M}\sum_{i=0}^{M-1} G_i(z)H_i(z),$$

which is the transfer function from $X(z)$ to $\hat{X}(z)$ when aliasing is canceled; the second term is made up of all aliasing terms (i.e. the terms that multiply $X(zW_M^k)$ for $k \neq 0$), which should all be identically zero in order to cancel aliasing.

In the best cases, *perfect reconstruction* can be achieved, where $\hat{x}[n]$ is a (possibly delayed and scaled) reproduction of the input $x[n]$.

4.4 Scalar Quantization and Entropy Coding

Quantization is the process of representing a value with reduced precision. The precision of the quantizer (or equivalently the amount of quantization noise) depends primarily on the spacing between quantization levels.

By quantizing the signals $y_i[m]$ at the output of the downsamplers with different precisions, one can achieve compression.

Let us imagine that we use a uniform quantizer with step-size Δ on each subband signal $y_i[m]$. In this case, each signal will incur a dirstortion of approximatly $\Delta^2/12$, in the absence of clipping [2]. One can follow each quantizer by a so-called *entropy coder*, whose role is to represent the quantized sequence of values $Q(y_i[m])$ with as few bits as possible. In order to do this, the entropy coder usually represents each possible value of the quantization index $Q(y_i[m])$ with a different number of bits. More specifically, the entropy coder will use short binary representations for values that occur often, and longer binary representations for values that do not occur often, thereby decreasing the average length needed to represent the data stream. For example, let us imagine that there are 4 possible output values in the quantizer, represented by v_1, v_2, v_3 and v_4. If these outputs have different probabilities of occurence as defined in table 4.1, then one can devise a variable length code to minimize the average length needed to represent these values, as shown in the third column of table 4.1, where the average number of bits to represent one of 4 symbols is now 1.5 bits, as compared to 2 bits if a natural binary numbering was used.

It can be shown that such an entropy code can be designed so that its average length (average number of bits per symbol) is arbitrarily close to the *entropy* H of the quantized values, defined

Symbol	Probability	Binary representation
v_1	0.2	01
v_2	0.1	001
v_3	0.05	000
v_4	0.65	1

Table 4.1 Example of an entropy code for the binary representation of a 4-symbol source.

as

$$H = \sum_i -p(v_i) \log_2 p(v_i),$$

where the sum runs over all possible values v_i of the source. For instance, for the 4-symbol source above, the entropy is given by

$$H = -0.2 \log_2 0.2 - 0.1 \log_2 0.1 - 0.05 \log_2 0.05 - 0.65 \log_2 0.65 = 1.4166 \text{ bits}.$$

In this experiment, we will not design the entropy coder, but instead evaluate the entropy of each quantized sequence, so that we have a lower-bound on the rate (in bits per sample) achievable for the coding of each subband signal.

The quantizers we will consider have a constant step-size Δ, and a potentially infinite dynamic range (i.e. the quantizer output index can be any integer value). In practise, the dynamic range of each subband signal will limit the efective dynamic range.

4.5 Preparation

- Derive in detail the z-domain expression of the input-output relationship of the analysis/synthesis system as in equation (4.2)
- Create a MATLAB function to estimate the entropy of a sequence of discrete values. This means that you first will have to evaluate the probabilities of every value in the sequence, and then compute the entropy H in bits.

4.6 Implementation

- The files `filtersLOT.mat` and `filtersCMFB.mat` contain the analysis filters (variable `H`) and the synthesis filters (variable `G`) for two different filter banks, in a matrix with one filter impulse response per row. Plot the filters in time and frequency .
- Implement the analysis and synthesis parts of your filter banks. Check with a test signal whether you have perfect reconstruction for the two sets of filters. What is the corresponding delay ?

- Implement quantizers between the analysis and synthesis parts of your filter bank. You should not use the MATLAB built-in functions for quantization, as they do not support an infinite dynamic range.
- Use your subband compression system to compress your test audio files at different rates (between 0.2 bps and 4 bps). In order to evaluate the rate for a given value of the quantizer step-size Δ, you should evaluate the entropy of each subband signal. In order to vary the bitrate, you should vary the step-size Δ. For each of the compressed signals, compute the SNR by a reference to the original. Also try a direct quantization of the time-domain signal to get different SNR values in the same bitrate range. Compare your results by plotting curves of SNR versus bitrate for different audio signals, subband or direct quantization, and the two different sets of filters.
- Study the repartition of the entropy between different subbands. Try to match this with the contents of the signal, as observed from a spectrogram.

4.7 A few hints

- You will be processing a lot of data. Carefully write your MATLAB code so as to vectorize as much as possible most of your operations.
- In order to load the audio files, use the MATLAB commands `auread` or `wavread`, depending on the file type. Note that some of your files have a stereo recording . They will give you a matrix with two rows or two columns. Select one of the channels for your processing, leaving out the other. In order to play a sound from MATLAB, use the command `sound`. Bring headphones to the lab in order to listen to samples.
- Make sure your function to estimate the entropy is efficient. In order to speed up your simulations, you might want to compile your function using the MATLAB compiler command `mcc -x` *functionname*.

References

[1] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing*. Prentice-Hall, third ed., 1996.

[2] B. Champagne and F. Labeau, “Class notes for the course ecse-412.” available on the web at `http://www.ece.mcgill.ca/courses/304-412A/Protected/main.pdf`.