Logarithmic Cubic Vector Quantization

Christian ROHLFING

Institute of Communication Systems and Data Processing, RWTH Aachen University, 52074 Aachen, Germany
rohlfing@ind.rwth-aachen.de

Abstract. In this paper we present “Logarithmic Cubic Vector Quantization” (LCVQ), a novel type of gain-shape vector quantization (GSVQ). In LCVQ, the vector to be quantized is decomposed into a gain factor and a shape vector which is a normalized version of the input vector. Both components are quantized independently and transmitted to the decoder.

Compared to other GSVQ approaches, in LCVQ the input vectors are normalized such that all shape vectors are located on the surface of the unit hypercube. As a conclusion, the shape vector quantizer can be realized based on uniform scalar quantizers. This yields low computational complexity as well as high memory efficiency even in case of very high input vector dimensions.

In order to demonstrate the coding efficiency of the proposed quantization scheme, LCVQ is compared to existing quantization schemes: A generalization of gain-shape vector quantization is given in Section 2 as well as brief descriptions of logarithmic spherical vector quantization (LSQ), LSVQ and the novel LCVQ.

Section 3 deals with the quantization scheme adaptive quantization backward (AQB). It shows interesting similarities to the previously mentioned GSVQ techniques LSVQ and LCVQ.

The quantization performance of all quantization techniques are evaluated in Section 4. Finally, conclusions are given in Section 5.

Keywords
Vector Quantization, Gain-shape Vector Quantization, Logarithmic Quantization.

1. Introduction

The quantizer is in general the key element in compression schemes for lossy speech and audio coding. In the design of a quantizer in practice, a good trade-off between complexity and quantization performance has to be achieved. On the one hand, scalar quantizers can be realized with low complexity but have only a moderate quantization performance. On the other hand, vector quantization techniques show promising results close to the theoretically achievable performance but have larger implementation costs. Recently, a gain-shape vector quantization (GSVQ) technique, logarithmic spherical vector quantization (LSQ), has been proposed which is a reasonably well-balanced solution to this problem [1]. Internally, it requires a vector codebook which is computed in an offline manner [2].

“Logarithmic Cubic Vector Quantization” (LCVQ) is a novel type of gain-shape vector quantization which offers some advantages over LSVQ: It operates with low complexity and high memory efficiency even in case of very high input vector dimensions. Therefore, it is flexible regarding the possibility to adjust the bit rate at runtime. It allows higher input vector dimensions than LSVQ, since no vector codebook is required.

In this paper, LCVQ shall be investigated and compared to other quantization schemes: A generalization of gain-shape vector quantization is given in Section 2 as well as brief descriptions of logarithmic spherical vector quantization (LSQ), LSVQ and the novel LCVQ.

Section 3 deals with the quantization scheme adaptive quantization backward (AQB). It shows interesting similarities to the previously mentioned GSVQ techniques LSVQ and LCVQ.

The quantization performance of all quantization techniques are evaluated in Section 4. Finally, conclusions are given in Section 5.

2. Generalized Gain-shape Vector Quantization

In gain-shape vector quantization (GSVQ) [3], the input vector \( x \in \mathbb{R}^L \) with dimension \( L \) is decomposed into a gain factor \( g \geq 0 \) and a shape vector \( c \in \mathbb{R}^L \) which are then quantized independently by means of a scalar quantizer for \( g \) and a vector quantizer for \( c \) and transmitted to the decoder (refer to Figure 1).

The two components are computed as

\[
g = \|x\|_p = \left( \sum_{i=1}^{L} x_i^p \right)^{\frac{1}{p}} \quad \text{and} \quad c = \frac{1}{g} \cdot x
\] (1)

with \( \|x\|_p \) denoting the \( p \)-norm of a vector.

In the decoder, combining the quantized version of the gain factor, \( \tilde{g} = Q_{g}(g) \) and of the shape-vector, \( \tilde{c} = Q_{c}(c) \) results in the overall reconstruction vector

\[
\tilde{x} = \tilde{g} \cdot \tilde{c}.
\] (2)

The main difference between LSVQ and LCVQ can be demonstrated based on equation 1:
In LSVQ, setting $p = 2$ results in a normalization of input vector $\mathbf{x}$ to its Euclidean norm

$$g = \|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^T \cdot \mathbf{x}}.$$  

(3)

In contrast to this, LCVQ makes use of the uniform (maximum) norm with $p = \infty$ instead of the Euclidean norm:

$$g = \|\mathbf{x}\|_\infty = \max(|x_1|, \ldots, |x_L|).$$  

(4)

Using the Euclidean norm for normalization yields in the shape vectors $\mathbf{c}$ being located on the surface of the $L$-dimensional unit hypersphere. In contrast to this, normalizing with the unit norm results in $\mathbf{c}$ lying on the surface of the $L$-dimensional unit hypercube. The difference is illustrated in Figure 2.

![Fig. 1. Parallel quantization of the gain and shape component with $p = 2$ and $Q_c = Q_{lsqv}$ for LSVQ and $p = \infty$, $Q_c = Q_{svq}$ for LCVQ.](image)

2.2. Logarithmic Spherical VQ (LSVQ)

In logarithmic spherical vector quantization, the Euclidean norm with $p = 2$ is used in equation 1 for normalization of input vector $\mathbf{x}$:

$$g = \|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^T \cdot \mathbf{x}}.$$  

(7)

In that case, the normalized shape vectors $\mathbf{c}$ are lying on the surface of the $L$-dimensional unit hypersphere.

The shape vector $\mathbf{c}$ is quantized using a spherical vector quantizer (SVQ) denoted by $Q_c = Q_{svq}$ whose codebook consists of $N_{svq}$ spherical codewords which are located on the surface of the unit sphere. The overall design goal of the codebook of $Q_{svq}$ is to distribute the codewords uniformly over the unit sphere surface.

[2] shows, that using LSVQ for quantization of the gain factor $g = \|\mathbf{x}\|_2$ and SVQ for the shape vector $\mathbf{c}$ results in a SNR which is independent of the PDF of the input vector:

$$\text{SNR}_{\text{lsqv}}(I) = \frac{N^2_{\text{svq}}}{C_{\text{svq}}} \cdot \left(\frac{\Gamma\left(\frac{L}{2}\right)}{2 \cdot (1 + \ln(A))}\right)^\frac{2}{L} \cdot \int_{\mathbf{x} \in \mathbb{R}^L} \int_{\mathbf{c} \in \mathbb{R}^L} p(\mathbf{x}) \cdot \|\mathbf{c}\|^2_2 \, d\mathbf{x}.$$  

(8)

Equation 8 is a qualitative formula for the SNR which can be derived from the normalized quantizer point density function under high bit rate assumptions. $C_{\text{svq}}$ denotes an unknown quantization cell form-factor and $\Gamma(z) = \int_0^\infty e^{-t} \cdot t^{z-1} \, dt$ the gamma-function [6].

High bit rate approximations yield another more quantitative formula for the SNR in dB as

$$\text{SNR}_{\text{lsqv}}(I)_{\text{dB}} = 6.02 \cdot R_{\text{eff}} - 10 \log_{10} \left(\frac{L}{(L + 1)^{2.5}} \cdot \frac{2 \cdot \sqrt{\pi} \cdot \Gamma\left(\frac{L+1}{2}\right)}{\Gamma\left(\frac{L}{2}\right)}\right)^\frac{2}{L} \cdot \left[\frac{(1 + \ln(A))^2}{12}\right]^\frac{1}{2}.$$  

(9)

The signal-to-noise ratio (SNR) of LQ is independent of the input vector probability density function (PDF)

$$\text{SNR}_{\text{log-lq}}(I)_{\text{dB}} = 6.02 \cdot R + 10 \log_{10}(3) - 20 \log_{10}(1 + \ln(A))$$  

(6)

with $R$ denoting the bit rate of the quantizer. This independence comes at the price of a penalty-term of $10 \log_{10}(3) - 20 \log_{10}(1 + \ln(A))$ compared to the well-known 6 dB-per-bit-rule [5].

As LQ can be considered as a special case of LSVQ and LCVQ for input vector dimension $L = 1$, it is used in Section 4 as a reference.

2.1. Logarithmic Quantization of the Gain Factor

In LSVQ and LCVQ, the gain factor $g$ is quantized using a logarithmic scalar quantizer (LSQ) $Q_g$. Here, the A-law companding algorithm shall be used with the number of reconstruction levels $N_g$, compression factor $A$ and stepsize $\Delta g$, as described in [4]. It can be shown, that in the case of LSVQ, the stepsize scales with the reconstruction value $\tilde{g}$ [5]:

$$\Delta g(\tilde{g}) = \frac{1 + \ln(A)}{N_g} \cdot \tilde{g}.$$  

(5)
This formula is used in Section 4 to compare the performance of LSVQ with the novel LCVQ scheme.

With $N_g$ being the number of quantization reconstruction levels of the gain-component, the total number of LSVQ codevectors is

$$N_{\text{lsvq}} = 2^{R_{\text{eff}} L} = N_g \cdot N_{\text{svq}}$$

with $R_{\text{eff}}$ denoting the effective bit rate per vector dimension. Naturally, a bit allocation is needed to distribute the overall bit rate optimally over the gain and the shape quantizer’s bit rates [2].

Figure 3 shows the overall LSVQ reconstruction values and quantization cells for the example of $L = 2$. The SVQ codevectors are located on versions of the shells of the unit sphere which are scaled by the LSQ reconstruction levels.

The design of spherical vector-codebooks used for $Q_{\text{svq}}$ is rather complex and shall not be discussed in this paper. Different procedures are given in [7] and [2].

2.3. Logarithmic Cubic VQ (LCVQ)

In “Logarithmic Cubic Vector Quantization”, the input vector is normalized by its uniform norm

$$g = ||x||_{\infty} = \max(|x_1|, \ldots, |x_L|) = x_{l_0}$$

with $l_0 = \arg \max_{l} |x_l|$ denoting the index of the maximum value of $x$ such that the normalized vector component $c_{l_0}$ = 1.

As a result, the shape vectors $c$ are lying on the surface of the $L$-dimensional unit hypercube with edge length equal to 2 as depicted in Figure 2(b).

The quantizer of the shape vector $Q_c = Q_{\text{cvq}}$ can be realized in a very efficient way as it consists of $L - 1$ uniform scalar quantizers $Q_{\text{sq}}$ for each dimension with index $l \neq l_0$

$$\tilde{c}_l = \begin{cases} Q_{\text{sq}}(c_l), & \text{if } l \neq l_0 \\ 1, & \text{if } l = l_0. \end{cases}$$

Each uniform scalar quantizer $Q_{\text{sq}}$ has the same number of reconstruction levels $N_c$ and the stepsize

$$\Delta c = \frac{2}{N_c}.$$ 

The total number of LCVQ reconstruction values is calculated as

$$N_{\text{lcvq}} = 2^{R_{\text{eff}} L} = N_g \cdot N_{\text{cvq}} = N_g \cdot 2 \cdot L \cdot N_c^{L-1}$$

with $N_{\text{cvq}}$ denoting the number of reconstruction values per cube shell. The factor $2 \cdot L$ gives the number of surfaces of a $L$-dimensional hypercube.

The overall LCVQ reconstruction values and quantization cells for vector dimension $L = 2$ are depicted in Figure 4. The reconstruction values are lying on cube shells which are scaled by the LSQ reconstruction levels and are equally distributed over the hypercube surface (similar to the SVQ codevectors, refer to Figure 3).

Using only scalar quantizers, it becomes clear that LCVQ offers the previously mentioned advantages such as low memory and implementation cost. Note that LCVQ has no limits regarding the input vector dimension in comparison to LSVQ, where the vector dimension is bounded by the codebook design.

3. Adaptive Quantization Backward (AQB)

In adaptive quantization backward, which was proposed in [9], the quantization stepsize $\Delta x$ is dynamically
adapted to the instantaneous signal power $\tilde{\sigma}_x$ of the quantized input signal $\tilde{x}(k)$ with $k$ being the current time index. $\tilde{\sigma}_x$ is estimated recursively using the latest transmitted value $Z(k - 1)$:
\[
\frac{\Delta x(k)}{\Delta x(k - 1)} = \frac{\tilde{\sigma}_x(k)}{\tilde{\sigma}_x(k - 1)} = f\{Z(k - 1)\} = M(k - 1).
\]
(15)

$M(k)$ denotes the so-called stepsize multiplier which is a function of $Z(k)$. $M(k)$ can be precalculated and stored in a table.

Figure 5 shows a block diagram of the AQB method: $Q_{sq}$ is a common uniform scalar quantizer with number of reconstruction levels $N_{sq} = 2^R$ and stepsize $\Delta x(k)$. $Q_{sq}$ provides $Z(k)$ as an intermediate value at time index $k$ which is also transmitted to the decoder. In the decoder, $Z(k)$ is transformed by $Q'_{sq}$ to the quantization reconstruction level $\tilde{x}(k)$. The stepsize-adaptation block is the same in encoder and decoder and uses equation 15 to estimate the instantaneous stepsize $\Delta x(k)$.

The AQB shall be considered as a reference quantizer in Section 4 as it shows similarities to LCVQ and LSVQ in the sense that all these algorithms have in common to estimate the instantaneous signal variance: AQB uses a direct estimation of the variance, whereas LCVQ and LSVQ are adapting to the variance by normalization of the input vector to the gain factor.

4. Evaluation

For the evaluation, a large number of signal vectors was generated following a normal or uniform distribution. These vectors were quantized afterwards with an LCVQ or an AQB quantizer for different vector dimensions $L \in \{2, 8, 48\}$. Finally, the signal-to-quantization-noise ratio (SNR) was calculated for each quantization scheme. The optimal LCVQ bit allocation for $N_p$ and $N_c$ was determined experimentally. The simulated SNRs of LCVQ and AQB are compared to the theoretical SNRs of LSQ (equation 9) and LSQ (equation 6) for different bit rates $R_{eff} \in [1.5, 5]$.

The companding factor $A$ is set to 5000 for all logarithmic quantizers, which was found in [10] as a reasonable trade-off between dynamic range and performance for audio signals.

Figure 6 shows the results of LCVQ compared to LSVQ. AQB and LSQ for uniformly and Figure 7 for normally distributed input vectors. LCVQ achieves almost the same SNR as LSVQ for the uniform input vector distribution for all input vector dimensions $L$. LCVQ shows better results than AQB for high bit rates and above vector-dimension $L = 8$.

For normally distributed input vectors, LSVQ outperforms LCVQ for all vector-dimensions $L$, whereby the difference between the LSVQ and LCVQ SNR increases with $L$. For vector-dimension $L = 8$ and lower bit rates, LCVQ provides a similar performance compared to AQB and performs better for high bit rates.

The worst SNR results in both PDF scenarios are obtained by LSQ which is the special case of LSVQ and LCVQ for vector dimension $L = 1$. 
LCVQ quantization cells introduce a larger quantization error due to their square shape compared to the non-square shaped LSVQ cells [8]. The overall square configuration of LCVQ reconstruction values is not optimal for spherically distributed sources either [2].

In practice, it is not feasible to generate LSVQ code-vectors above a vector dimension of $L = 16$ whereas LCVQ is still able to quantize vectors with very large dimensions. This is one of the key advantages of LCVQ compared to LSVQ.

5. Conclusions

In this paper, a novel gain-shape vector quantization technique, “Logarithmic Cubic Vector Quantization”, was proposed and compared to other quantization schemes, particularly to logarithmic spherical vector quantization. LCVQ offers the advantages of low complexity, high memory efficiency and adaptivity to different bit rates at runtime. LSVQ shows slightly better SNR results than LCVQ. By a significant increase of the input vector dimension, which is not possible for LSVQ, LCVQ can overcome the performance loss compared to LSVQ.

Acknowledgements

Research described in the paper was supervised by Dr.-Ing. H. Krüger and Prof. Dr.-Ing. P. Vary from the Institute of Communication Systems and Data Processing, RWTH Aachen University.

References


About Authors...

Christian ROHLFING was born in Krefeld, Germany. He began his studies of Computer Engineering at RWTH Aachen University in 2006 and is currently writing his diploma thesis at the Institute of Communication Systems and Data Processing, RWTH Aachen University.