6-1-1999

Novel Quantization Schemes for Very Low Bit Rate Video Coding Based on Sample Adaptation

Dong Sik Kim  
*Purdue University School of Electrical and Computer Engineering*

Ness B. Shroff  
*Purdue University School of Electrical and Computer Engineering*

Follow this and additional works at: [http://docs.lib.purdue.edu/ecetr](http://docs.lib.purdue.edu/ecetr)

[http://docs.lib.purdue.edu/ecetr/42](http://docs.lib.purdue.edu/ecetr/42)

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
NOVEL QUANTIZATION SCHEMES FOR VERY LOW BIT RATE VIDEO CODING BASED ON SAMPLE ADAPTATION

DONG SIK KIM
NESS B. SHROFF

TR-ECE 99-9
JUNE 1999

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285
Novel Quantization Schemes for Very Low Bit Rate Video Coding Based on Sample Adaptation

Dong Sik Kim    Ness B. Shroff
School of Electrical and Computer Engineering
1285 Electrical Engineering Building
Purdue University
West Lafayette, IN 47907-1285

1This work has been supported in part by LG Electronics through grant 4049670, and the National Science Foundation through grant NCR-9624525.
## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Sample-Adaptive Product Quantization: Asymptotic Analysis and Examples</td>
<td>3</td>
</tr>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Sample-Adaptive Product Quantization</td>
<td>4</td>
</tr>
<tr>
<td>Performance of SAPQ</td>
<td>6</td>
</tr>
<tr>
<td>Examples in Designing SAPQ</td>
<td>10</td>
</tr>
<tr>
<td>Conclusion</td>
<td>19</td>
</tr>
<tr>
<td>DCT Coefficient Quantization Using Non-Zero Level Codebooks based on</td>
<td>21</td>
</tr>
<tr>
<td>the Sample-Adaptive Product Quantization</td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>21</td>
</tr>
<tr>
<td>Uniform Sample-Adaptive Product Quantization</td>
<td>22</td>
</tr>
<tr>
<td>Adaptive Quantizer Scheme for H.263+</td>
<td>24</td>
</tr>
<tr>
<td>Quantizer Design and Simulation</td>
<td>26</td>
</tr>
<tr>
<td>Conclusion</td>
<td>33</td>
</tr>
<tr>
<td>Generalized SAPQ for k-Dimensional Vectors</td>
<td>37</td>
</tr>
<tr>
<td>Generalization of SAPQ</td>
<td>37</td>
</tr>
<tr>
<td>SAPQ for Gauss-Markov Sources</td>
<td>38</td>
</tr>
<tr>
<td>Image Vector Quantization</td>
<td>38</td>
</tr>
<tr>
<td>Performance Comparison and Trellis Coded Quantization</td>
<td>41</td>
</tr>
<tr>
<td>Differential Sample-Adaptive Product Quantizer</td>
<td>49</td>
</tr>
<tr>
<td>Introduction</td>
<td>49</td>
</tr>
<tr>
<td>Differential Sample-Adaptive Product Quantizer</td>
<td>50</td>
</tr>
<tr>
<td>DSAPQ Design Based on Clustering</td>
<td>52</td>
</tr>
<tr>
<td>Numerical Results</td>
<td>55</td>
</tr>
<tr>
<td>Discussions on the Generalized SAPQ and DSAPQ</td>
<td>56</td>
</tr>
<tr>
<td>Conclusion</td>
<td>59</td>
</tr>
<tr>
<td>Robustness of SAPQ in Channel Noise</td>
<td>63</td>
</tr>
<tr>
<td>Introduction</td>
<td>63</td>
</tr>
<tr>
<td>Channel-Limit Performance</td>
<td>64</td>
</tr>
<tr>
<td>Error Performance of SAPQ</td>
<td>65</td>
</tr>
<tr>
<td>Image Vector Quantization</td>
<td>66</td>
</tr>
<tr>
<td>Conclusion</td>
<td>71</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

2.1 Block diagram of SAPQ in Example 2.1 (Lattice $D_m^A$). ........................................ 11
2.2 The codebooks of SAPQ in m-dimensions for different initial guess (Gaussian i.i.d., input with
the variance 1, $n = 2$, $m = 2$, and $\eta = 1$. Note that each codebook is the union of two product
codebooks.). (a) Initial Guess 1 (distortion: -6.93 dB). (b) Initial Guess 2 (distortion: -6.14 dB). 14
2.3 The codebooks of SAPQ in m-dimensions for different initial guesses (Gaussian Markov-1 source
with the variance 1 and the correlation coefficient 0.9, $n = 2$, $m = 2$, and $\eta = 1$). (a) Split method
with $\varepsilon = 0.01$ (distortion: -8.07 dB). (b) Split method with $\varepsilon = 1$ (distortion: -8.73 dB). .... 15
2.4 Distortions (dB) of SAPQ and the codebook-constrained SAPQ for different values of $\eta$. (Gaussian
i.i.d., input with the variance 1 and the ratio $m/n = 1$). ................................................. 16
2.5 The codebook-constrained SAPQ and PQ, $n = 16$ and the results are obtained by varying $m$
($m = 8, 16, 24$), for each $\eta$. (a) Gaussian i.i.d. with the variance of 1. (b) Laplacian i.i.d.
with the variance of 1. ............................................ 17

3.1 Lattice $D_m^A$ based uniform SAPQ ($\eta = 1$) distortion (dB) with respect to the sample size $m$.
($D_{SLB}$ is Shannon's low bound given by $(6/\pi)e^{2R}$). .............................................. 23
3.2 Number of coded blocks in each frame of a QCIF video sequence, FOREMAN at QP = 16.
(TH = 10: Y-blocks that requires more or equal to 10 bits, and TH = 20: Y-blocks that requires
more or equal to 10 bits.) ............................................. 27
3.3 Quantization reconstruction levels of IQ, in the SAPQ. (IQ, for $i = 1, \ldots, m$ in the SAPQ based on
CR-$D_m^A$) ............................................. 28
3.4 8 x 8 DCT block and the 8 coefficients for SAPQ ($m = 8$). ............................................ 29
3.5 Bits/Frame Comparison (Foreman QCIF, QP = 8). (a) TMN8 (33.06 dB and 4916
bits/frame) and Modified TMN8 (QP-1 for Busy Y-Blocks, 33.40 dB, and 5364 bits/frame)
(b) TMN8, AQS($\eta = 2, m = 6, 33.10$ dB, and 5032 bits/frame), and AQS($\eta = 4, m = 8,
33.16$ dB, and 5086 bits/frame) ............................................. 33
3.6 Frame 10 from the reconstructed FOREMAN QCIF video sequence. (a) TMN8 at QP = 16. (b)
TMN8 at QP = 15. (c) AQS at QP = 16 and $\eta = 2$. (d) AQS at QP = 16 and $\eta = 4$. ........ 34
3.7 Frame 19 from the reconstructed FOREMAN QCIF video sequence. (a) TMN8 at QP = 16. (b)
TMN8 at QP = 15. (c) AQS at QP = 16 and $\eta = 2$. (d) AQS at QP = 16 and $\eta = 4$. .... 35
3.8 8 x 8 DCT block and the 6 coefficients for SAPQ ($m = 6$). ............................................ 36
3.9 AQS at QP = 16, $\eta = 2$, and $m = 6$. (a) Frame 10. (b) Frame 19. ................................. 36

4.1 Improvement of performance by increasing k for a Gaussian i.i.d. and a Gauss–Markov source
(1st order Markov with the correlation coefficient 0.9). (1-SAPQ results are obtained by varying
$\eta$ from 1 to 4, where the split constant $\varepsilon = 0.001$, otherwise specified.) .......... 39
4.2 Ten 512 x 512 images with 8 b/pixel for TS (From the left upper image, Peppers, Airforcebase,
Cablecar, Crowd, Diving, Einstein, Fruit, Lighthouse, Baboon, and Trains). ................. 40
4.3 Image subblocks for 1-SAPQ. (a) $k = 2 \times 2$ and $m = 2 \times 2$. (b) $k = 2 \times 2$ and $m = 4 \times 1$. .... 40
4.4 Original images (512 x 512, 8 b/pixel). (a) Lena. (b) Bridge. .................................... 42
4.5 Reconstructed images at 1.0 b/pixel. (a) Lena, VQ with $k' = 2 \times 2$ (29.8 dB). (b) Lena, SAPQ
with $k = 2 \times 2$ and $m = 2 \times 2$ (31.8 dB). (c) Bridge, VQ with $k' = 2 \times 2$ (25.5 dB), (d) Bridge,
SAPQ with $k = 2 \times 2$ and $m = 2 \times 2$ (26.6 dB). ............................................. 43
4.6 Comparison of SAPQ and VQ ($\eta = m/2$). (a) Number of codewords for codebook-constrained
SAPQ and VQ. (b) Distortion (dB) of codebook-constrained SAPQ and VQ for a Gaussian i.i.d.
with unit variance. ............................................. 45
4.7 Distortions (dB) for various coding schemes for a Gaussian i.i.d. source with unit variance. (DRF: distortion rate function, TCQ Opt.: TCQ using optimized alphabets [34], TCQ: TCQ using rate-
(R + 1) Lloyd-Max [34], Ent.Const.: entropy constraint optimum [25], Ent.Cod.: entropy-coded quantization, Lloyd-Max with Huffman [8], I-SAPQ Ent.: I-SAPQ with respect to its entropy.
1-SAPQ results are obtained by varying q = 1, · · · , 5, for each k and n', and Trellis size in TCQ and quantizer dimension km in I-SAPQ are equal to 16.) ....................................................... 46

5.1 Block diagram of DSAPQ. (a) SAPQ part (N = 2^n). (b) DQj: the jth DPCM part with a
distortion accumulator for DSAPQ (j = 1, · · · , N). ................................................. 53

5.2 The codebook-constrained SAPQ and DSAPQ for the Gaussian pdf (n' = 16). (a) Codebook-
constrained SAPQ designed for Gaussian i.i.d. with \( \sigma^2 = 1 \). (b) DSAPQ for Gaussian Markov-1
with \( \sigma^2 = 1 \) and \( \rho = 0.9 \) (24 = \( p \cdot 2^{l-1} \)). .................................................. 54

5.3 The codebook-constrained SAPQ and DSAPQ for the Laplacian pdf (n' = 16). (a) Codebook-
constrained SAPQ designed for Laplacian i.i.d. with \( \sigma^2 = 1 \). (b) DSAPQ for Laplacian Markov-1
with \( \sigma^2 = 1 \) and \( \rho = 0.9 \) (\( t_{\ell} = p \cdot 2^{l-1} \)). .................................................. 54

5.4 Parts of the DPCM and DSAPQ coded images of Lena. (a) The original image. (b) DPCM
with \( n = 19 \) (\( R \approx 4.25 \) b/pixel, 40.50 dB). (c) DSAPQ with \( n' = 16 \), \( \eta = 2 \), and \( m = 8 \)
(\( R = 4.25 \) b/pixel, 41.95 dB). (d) DSAPQ with \( n' = 16 \), \( \eta = 4 \), and \( m = 16 \) (\( R = 4.25 \)
b/pixel, 43.95 dB). ................................................................. 57

5.5 2-dimensional predictor for image data: \( X = (A + B)/2 \). ........................................ 58

5.6 PSNR comparison between DPCM and DSAPQ for real images with respect to \( n' \) (2-dimensional predictor: \( X = (A + B)/2 \), \( m = 16 \), and \( \eta = 4 \). (a) Lena (Scale factor = 14). (b) Bridge (Scale
factor = 16). ................................................................. 58

5.7 Reconstructed images at 1.25 b/pixel (Lena). (a) \( k = 2 \times 2 \), \( m = 2 \times 2 \), \( n' = 16 \), and \( \eta = 16 \)
1-SAPQ (33.0 dB). (b) \( k = 1 \), \( m = 16 \), \( n' = 2 \), and \( \eta = 16 \) DSAPQ with 2-dimensional predictor
(32.7 dB). ................................................................. 60

5.8 Reconstructed images at 1.25 b/pixel (Bridge). (a) \( k = 2 \times 2 \), \( m = 2 \times 2 \), \( n' = 16 \), and \( \eta = 16 \)
1-SAPQ (27.9 dB). (b) \( k = 1 \), \( m = 16 \), \( n' = 2 \), and \( \eta = 16 \) DSAPQ with 2-dimensional predictor
(28.5 dB). ................................................................. 60

5.9 DPCM with the 2-dimensional predictor at 1 b/pixel. (a) Lena (25.8 dB for scale factor = 14).
(b) Bridge (23.3 dB for scale factor = 16). ................................................................. 61

5.10 DPCM with the 2-dimensional predictor at 2 b/pixel. (a) Lena (33.0 dB for scale factor = 14).
(b) Bridge (30.1 dB for scale factor = 16). ................................................................. 61

6.1 Distortion (dB) of quantizers with respect to bit error rate, and channel-limit performance. (a)
Scalar quantizers (m = 1) for various bit rate (NBC: natural binary code, FBC: folded binary
code). (b) Channel-limit performance, C1, C2, and C3 for SQ (k' = 1), VQ (k' = 2), and VQ
(k' = 4), respectively. ................................................................. 64

6.2 Different quantization schemes and channel-limit performance, C1 for SQ (k' = 1) and SAPQ
(k = 1, m/n' = 4), and C2 for VQ (k' = 2). ................................................................. 66

6.3 Comparison of SAPQ (k = 1) and VQ (k' = 2) for the Gaussian i.i.d. source (C1: SAPQ, C2:
VQ). (a) SAPQ: m = 2 and \( \eta = 1 \). (b) SAPQ: m = 8 and \( \eta = 4 \). ................................................................. 67

6.4 Bit protection of SAPQ (k = 1, m = 8, \( n' = 16 \), \( \eta = 4 \)) and VQ (k' = 2, \( \nu' = 512 \)) for the
Gaussian i.i.d. source at R = 4.5. (a) Distortion with respect to bit protections from MSB. (b)
MSB protection. ................................................................. 67

6.5 Reconstructed images at 1.0 b/pixel and BER = \( 5 \times 10^{-3} \). (a) Lena, VQ with k' = 2 \times 2 (26.5
dB). (b) Lena, SAPQ with k = 2 \times 2 and m = 2 \times 2 (37.2 dB). (c) Bridge, VQ with k' = 2 \times 2
(23.9 dB). (d) Bridge, SAPQ with k = 2 \times 2 and m = 2 \times 2 (24.7 dB). ................................................................. 68

6.6 Comparison of SAPQ and VQ for real images at 1.0 b/pixel (VQ: k' = 2 \times 2 and \( \nu' = 16 \), SAPQ:
k = 2 \times 2, m = 2 \times 2, n' = 8, and \( \eta = 4 \)). ................................................................. 69
6.7 Reconstructed Lena (VQ+Huffman with \( k' = 2 \times 2 \) and \( \nu' = 32 \) at \( R = 1.1 \) b/pixel; PSNR = 31.4 dB).

6.8 Reconstructed images of Lena (VQ+Huffman; \( k' = 2 \times 2 \) and \( \nu' = 32 \) at \( R = 1.1 \) b/pixel, SAPQ: \( k = 2 \times 2, m = 2 \times 2, n' = 8, \) and \( \eta = 4 \) at \( R = 1.0 \) b/pixel). (a) VQ+Huffman at BER = \( 10^{-4} \) (PSNR = 23.5 dB). (b) VQ+Huffman at BER = \( 10^{-3} \) (PSNR = 12.0 dB). (c) SAPQ at BER = \( 10^{-4} \) (PSNR = 31.7 dB). (d) SAPQ at BER = \( 10^{-3} \) (PSNR = 30.4 dB).
Abstract

In this report, we introduce a novel feed-forward adaptive quantization scheme called SAPQ (sample-adaptive product quantizer) as a structurally constrained vector quantizer. SAPQ is based on a concept of adaptive quantization to the varying samples of the source and is very different from traditional adaptation techniques for non-stationary sources. SAPQ quantizes each source sample using a sequence of quantizers. Even when using scalar quantization in SAPQ, we can achieve performance comparable to vector quantization (with the complexity still of the order of scalar quantization). We also show that important lattice based vector quantizers can be constructed using scalar quantization in SAPQ with several examples. We asymptotically analyze SAPQ and propose a simple algorithm to implement it. We numerically study SAPQ for independent, and identically distributed Gaussian and Laplacian sources. Through our numerical study, we find that SAPQ using scalar quantizers achieves typical gains of $1 \sim 3$ dB in distortions over the Lloyd-Max quantizer. By employing SAPQ, we have extensively conducted image compressions. We considered a uniform quantizer for the current H.263 standard and a non-uniform quantizer for the differential pulse code modulation for images. We also show that a generalized SAPQ can be used in conjunction with vector quantizers to further improve the gains, especially for high correlated image signals at large vector dimensions. Finally, the error-resilient aspect of SAPQ is also investigated. We find that SAPQ is robust to the channel noise, while achieving gains.
1. Introduction

These days, the most successful and widely used video compression technique is the hybrid codec, which is a combination of motion-compensated prediction (MCP) and discrete cosine transform (DCT) schemes. The increasing hardware computational capability enables us to incorporate more sophisticated schemes into the codec systems to achieve better performance. For example in MCP, half-pixel motion estimation and bidirectional frames were first introduced in MPEG-1, and H.263 has adopted the variable block-size motion compensation and overlapped-block motion vectors. For the future H.26L standard, MCP may be extended to an affine motion field model and segmented image for arbitrary shaped regions. In the DCT case, the wavelet based coding scheme may be substituted for DCT in JPEG 2000 for still image coding, and multi-shaped DCT is also considered in H.26L for video codecs. Instead of using DCT, the matching pursuit technique has also been proposed for video compression [60],[1]. However, recently, research on employing the vector quantizer (VQ) is being conducted especially for very low bit-rate codecs [21],[61]. It is believed that the decorrelation efficiency of DCT for the MCP residual signal is lower than the VQ case for very low bit-rate codecs. Further, in order to achieve gains from the quantization part, the current trend is employing rate-distortion based optimization schemes. Hence, conducting research for using VQ techniques in future video codecs is necessary. Due to the rapid growth of computing power, it will be practical in exploiting the advantage of VQ.

The image signal is traditionally regarded as a composite source. In general, an image consists of low active (homogeneous) regions and high active (or complicate) regions. The homogeneous regions usually correspond to the background or texture regions, and are relatively large. On the other hand, the high active regions have correlation within a small area especially for MCP residual signals. Hence, it is hard to achieve a significant improvement by using a fixed block size DCT. Instead of such a DCT, employing different block sizes will yield an improvement over the fixed 8 x 8 image blocks [9]. Furthermore, introducing variable dimension VQ for the various block sizes can achieve gains over the traditional DCT based scalar quantization. In fact, the major role of DCT in the very low bit-rate codec is for finding non-coded blocks and data rearrangement for the scalar quantization followed by the run-length and multidimensional Huffman coder, rather than the decorrelation. On the other hand, VQ can achieve gains even from nonlinear correlation [16], and even if there were no correlations, using VQ we can obtain an improvement (from the source coding theorem [81]).

As an example of a coding scheme using various image block sizes, we now describe a quad-tree based variable dimension VQ [25],[78]. First, a segmentation algorithm is used to segment the low and high active areas. Regarding the segmentation algorithm for image compression, the quad-tree based tree representation scheme is very useful because of its low required side bits; the quad-tree representation has a modest overhead [73],[78]. In the quad-tree representation, each node has four children unless the node is terminal. The split and merge method may be appropriate for segmenting images [25],[61]. In each node, we could have different block sizes or vector dimensions, hence, we can apply different dimensional VQs or different types of VQs to each node. In the literature (including several H.26L proposals), we can find the block sizes, 32 x 32, 16 x 16, 8 x 8, 4 x 4, and 2 x 2 [78],[71],[79]. For each block size, we can use different coding schemes. The Nokia scheme uses DCT and entropy-constrained VQ (ECVQ) for 8 x 8 and 4 x 4 following some directional classifiers [61]. The SCT (Strathclyde compression transform) [21] uses four kinds of block sizes and each block size type has 256 codewords, which is based on the notion that small blocks should encode areas of high detail [71]. However, depending on the image block sizes, selecting the quantization type is quite heuristic. In fact, if we could implement high dimensional VQs such as 32 x 32 for relatively large bit-rates, then perhaps we do not need such complicated segmentation and variable dimensional VQ techniques. Unfortunately, it is nearly impossible to implement such 32 x 32-dimensional VQ at desired bit-rates, using the current techniques in the literature. However, if we use the structurally constrained VQ (described in this report), SAPQ, then we can design a VQ for such large image block sizes. Further our quantization scheme could be able to employ the optimization techniques based on the Lagrange multiplier considering all the different sized blocks [62].
Recently, another significant problem in multimedia communication is the error-resilient transmission of image and video data over noisy communication channels, especially mobile wireless channels and networks with packet loss. Note that the current quantization techniques, which are based on the scalar quantizer followed by the variable-length coder (VLC) suffer from the error propagation, data loss, and buffer control problems. Consequently, the main goal of this research can be summarized as follows.

- **Finding the best compression scheme for various sized image or residual image blocks using SAPQ:**
  - Developing appropriate SAPQ design algorithm for highly correlated signals, such as images.

- **Finding robust quantization techniques for image and video coding from the error-resilient aspects:**
  - Adopting SAPQ into the quantization of image and video signals with extensive error analysis.

In the following sections, we will describe our basic approach to achieve the goals by using the SAPQ technique. In this report, we will briefly describe some results under the assumption that the readers are experts in VQ.
2. Sample-Adaptive Product Quantization: Asymptotic Analysis and Examples

2.1 Introduction

Vector quantization (VQ) is an efficient data compression technique for low bit-rate applications (i.e., below 1 bit per point). By employing VQ, we can achieve high gains especially for image and speech/audio data. Image and speech/audio data are highly correlated, and cannot be decorrelated using conventional linear transforms, such as the discrete cosine transform. Depending on the input sources, using a combination of a scalar quantizer and an entropy coder, it is possible to obtain performance up to 1.533 dB lower than the theoretical bound. However, using VQ one can further improve this performance, and come closer to achieving the theoretical lower bound.

It is VQ's ability to improve upon scalar quantization that has lead future coding standardization efforts, such as JPEG 2000 (Joint Picture Experts Group 2000) and the ITU-T (International Telecommunication Union - Telecommunication Standardization Sector) H.26L, to consider using VQ for their quantization scheme. However, the major disadvantage of VQ is its encoding complexity, which increases dramatically with the vector dimension and bit-rate. This is especially problematic since current coding schemes are based on high bit-rate quantizers, even when the coding schemes are designed for a low bit-rate transmission system. Hence, applying VQ to current coding structures is very difficult. In order to circumvent this problem, various modified VQ techniques have been proposed [25], e.g., tree-structured VQ, classified VQ, and the lattice VQ [13],[44]. However, since such schemes are still based on a VQ structure, the application areas of these schemes are relatively limited. More recently, the trellis coded quantization (TCQ) schemes have gained popularity for their ability to provide high performance for lower complexity (than traditional VQ schemes). Unfortunately, since TCQ requires special techniques such as the trellis encoder, and the Viterbi decoder, implementing the TCQ based coding scheme is still quite complex.

In this section, we propose a feed-forward adaptive quantizer based on the structure of a scalar quantizer to obtain VQ-level performance. In traditional adaptive quantization schemes the adaptation period is long since the adaptation follows the varying statistical characteristics of the source [67],[39]. In contrast to these schemes we use a a short adaptation period for our quantization scheme. Hence, we call this scheme the sample-adaptive product quantizer (SAPQ). Further, this quantizer is a structurally constrained VQ, where the vector dimension is equal to the adaptation period. As we will demonstrate later, using SAPQ we can obtain a high gain over scalar quantizers even for independent, and identically distributed (i.i.d.) sources. For high bit-rates (greater than 1 bit per point), we can obtain VQ-level performance while maintaining the low encoding complexity of a scalar quantizer. This is an important achievement since as mentioned before, obtaining VQ-level performance for high bit-rates is usually quite difficult due to the required high encoding complexity. Moreover, our quantization scheme can be extended to the low bit-rate cases as well [45].

Another appealing quality of our proposed adaptive scheme is that the main idea is quite intuitive and simple, compared to TCQ. Further, the proposed quantization scheme can describe several lattice quantizers which allows for low complexity coding. In a coding scheme that does not employ the entropy coder for the quantizer output, the proposed quantizer can provide a 2–3 dB improvement over the Lloyd-Max scalar quantizers [38].

This section is organized as follows. In Section 2.2, we mathematically define the proposed quantizer, and analyze the asymptotic performance of the quantizer in Section 2.3. In Section 2.4, we introduce several design examples and simulation results with discussions. We then conclude the section in the last section.
2.2 Sample-Adaptive Product Quantization

In this section, we mathematically define the proposed quantization scheme and lay the foundation for the asymptotic analysis conducted in Section 2.3.

We consider a sequence of random variables $X_1, \ldots, X_m$ taking values in $\mathbb{R}$ as the discrete-time source to be quantized. Here $m$ is the adaptation period or sample size. Suppose that $E\{X_i^2\} < \infty$, $i = 1, \ldots, m$. Let $\mathcal{C}_n$ denote the class of sets that take $n$ points from $\mathbb{R}$, and let the sets in $\mathcal{C}_n$ be called "n-level codebooks", where each such codebook has $n$ codewords. The quantization of $X_i$ is the mapping of a sequence of observations of $X_i$ to a sequence of points of $\mathcal{C} (\in \mathcal{C}_n)$ according to a mapping called the quantizer. The average distortion achieved when a random variable $X_i$ is quantized by a codebook $\mathcal{C} (\in \mathcal{C}_n)$, is given by

$$E \left\{ \min_{y \in \mathcal{C}} (X_i - y)^2 \right\}. \quad (2.1)$$

In this quantization scheme, if fixed length binary codes are used to represent the quantizer outputs, the bit-rate $R$ (defined as bits per source point in $\mathbb{R}$) required is $R = \log_2 n$. Note that, since $X_i \in \mathbb{R}$ and $\mathcal{C} \subset \mathbb{R}$, the quantizer in (2.1) is a scalar quantizer.

**A. Sample-Adaptive Product Quantizer**

Let an observation of $X_1, \ldots, X_m$ be denoted by $x_1, \ldots, x_m$; we call this observation a sample. Suppose that the codebooks $\mathcal{C}_i$ are $\mathcal{C}_i \subset \mathbb{C}_n$ for $i = 1, \ldots, m$, where $n_i$ are positive integers. If we quantize this sample by applying scalar quantizers using codebooks $\mathcal{C}_i$ to each $x_i$ independently, the overall average distortion $D_{PQ}$ is given by

$$D_{PQ} := E \left\{ \frac{1}{m} \sum_{i=1}^{m} \min_{y \in \mathcal{C}_i} (X_i - y)^2 \right\}. \quad (2.2)$$

We call this quantization scheme the product quantizer (PQ), since the quantizer is a mapping from $\mathbb{R}^m$ to the product set $\mathcal{C}_1 \times \ldots \times \mathcal{C}_m$. The size of the product set or the codebook is $n = n_1 \ldots n_m$. Hence, the total bit-rate $R$ is given by

$$R = \frac{1}{m} \log_2 \prod_{i=1}^{m} n_i. \quad (2.3)$$

If the random variables are independent (or uncorrelated), then this independence appears to motivate quantizing each random variable independently as shown in (2.2). However, even if the input is independent, independently quantizing each random variables of $X_1, \ldots, X_m$ is just one of the many possible coding schemes and could be improved by appealing to the block source coding theorem [81]. If $\mathcal{C}$ is a subset of $\mathbb{R}^m$ with $|\mathcal{C}| = v$, where $v$ is a positive integer, then the average distortion yielded by using a vector quantizer for $X_1, \ldots, X_m$ is

$$D_{VQ} = E \left\{ \min_{y \in \mathcal{C}} \frac{1}{m} \sum_{i=1}^{m} (X_i - y_i)^2 \right\}, \quad (2.4)$$

where $y = (y_1, \ldots, y_m) (\in \mathbb{R}^m)$ and the bit-rate is $R = (\log_2 v)/m$. Note that, for fixed values of bit-rate $R$ and $m$,

$$\inf D_{VQ} < \inf D_{PQ}, \quad (2.5)$$

where the infimums of $D_{VQ}$ and $D_{PQ}$ are taken over all possible choices of codebooks $\mathcal{C}$ in $\mathbb{R}^m$ and the codebook $\mathcal{C} = \mathcal{C}_1 \times \ldots \times \mathcal{C}_m$, where the $m$ codebooks are $\mathcal{C}_i \subset \mathcal{C}_n$, in $\mathbb{R}$, respectively.
Now, we introduce a feed-forward adaptive quantization scheme, which is based on a new concept of adaptation to each sample of $X_1, \ldots, X_m$ (called sample adaptation). Sample adaptation, as we will soon see, allows us to improve upon the distortion in (2.2). Let $C_{i,j} \in \mathbb{R}$ denote the $i$th codebook for each $X_i$, where $j \in \{1, 2, \ldots, 2^q\}$, and $\eta \in \mathbb{N}$. The sample adaptive scheme quantizes each sample $x_1, \ldots, x_m$ using the codebooks $C_{1,j}, \ldots, C_{m,j}$ to form the $2^q$ candidates of distances

$$\frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i,j}} (x_i - y)^2, \text{ for } j = 1, 2, \ldots, 2^q,$$

and choose the smallest distance. Hence, the average distortion in the adaptive quantization scheme is given by

$$D_{\text{SAPQ}} := E \left\{ \min_j \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i,j}} (X_i - y)^2 \right\}. \quad (2.7)$$

Here, we suppose that $C_{i,j} \subset C_{i,1}$, for $j = 1, 2, \ldots, 2^q$, where $n_i' \in \mathbb{N}$. We call this quantization scheme, the sample-adaptive product quantizer (SAPQ). For each sample, the adaptive scheme produces the bit streams for a codebook index and the $m$ quantized element indices, in the form of the feed-forward adaptive coding scheme. This makes it possible to replace different codebooks for each sample of $X_1, \ldots, X_m$. Therefore, the total bit-rate is given by

$$R = \frac{1}{m} \log_2 \prod_{i=1}^{m} n_i' + \frac{\eta}{m}, \quad (2.8)$$

where $\eta$ are the additional bits (side information) required in our scheme to indicate which codebook is employed. Note that, throughout this section, we will suppose that $\log_2 n_i$ and $\log_2 n_i'$ can be non-integers for the quantizer performance comparisons. Although our discussion in this section has focused on $m$ random variables, we can also consider $m$ k-dimensional random vectors as the input of the SAPQ. This generalization of SAPQ will be described in the next section [46].

It is important to note that the proposed scheme is very different from traditional adaptive coding schemes that produce increased gains by replacing the quantizer depending on the varying statistical characteristics of a non-stationary source [67], [63]. We will now show that SAPQ is a structurally constrained VQ in m-dimensions.

B. SAPQ: A Structurally Constrained VQ

The codebook of PQ in m-dimensions is in the form of the product codebook $C_1 \times \ldots \times C_m$. Further, in the m-dimensional VQ, the codebook can be a set of any $n$ points in $\mathbb{R}^m$ (remember that $n$ is the size of a codebook). However, the codebook of SAPQ is the union of product codebooks, i.e., $U_{j=1}^{2^q} (C_{i,j} \times \ldots \times C_{m,j})$, where $2^q$ is the number of the product codebooks and $\prod_{i=1}^{m} n_i' = |C_{1,j} \times \ldots \times C_{m,j}|$ is the size of a product codebook. Hence, it should be clear that, for the same bit-rate, the performance of SAPQ is always better than or equal to PQ in (2.2). In other words, when $R = (\log_2 \prod_{i=1}^{m} n_i')/m = (\log_2 \prod_{i=1}^{m} n_i')/m + \eta/m$ the distortion of SAPQ is always less than the distortion of PQ, i.e.,

$$\inf D_{\text{SAPQ}} \leq \inf D_{\text{PQ}}, \quad (2.9)$$

where the infimum of $D_{\text{SAPQ}}$ is taken over all choices of the codebook $U_{j=1}^{2^q} (C_{i,j} \times \ldots \times C_{m,j})$, and where the $2^q$ codebooks are $C_{i,j} \subset C_{i,1}$ in $\mathbb{R}$. Note that this relation can be proven by investigating the codebook structure of SAPQ and PQ in m-dimensions. Since the adaptive quantizer is a structurally constrained m-dimensional VQ, the performance of the adaptive quantizer is between that of the scalar quantizer and the VQ in m-dimensions. In other words,

$$\inf D_{\text{VQ}} \leq \inf D_{\text{SAPQ}}. \quad (2.10)$$
However, in the next section, it will be shown that SAPQ can asymptotically achieve the m-dimensional VQ performance. It is important to note that the encoding complexity of SAPQ is on the order of the encoding complexity of a scalar quantizer (for a detailed discussion on encoding complexity of various quantizers, see [45]). A discussion on the encoding complexity in conjunction with the necessary memory requirements will be given in Section 2.4.

2.3 Performance of SAPQ

In this section, through asymptotic analysis, we will formally study the performance of SAPQ.

A. Asymptotic Analysis

We will continue to view SAPQ as a structurally constrained VQ in m-dimensions. To simplify the notation, let \( X := (X_1, \ldots, X_m) \) denote an m-dimensional random vector. We assume an absolutely continuous distribution function for \( X \).

Now consider root lattices [37]. Let the points of an m-dimensional lattice \( \mathcal{L}_m \subset \mathbb{R}^m \) be denoted by \( y_j, j \in \mathbb{Z} \). The closure of the ith Voronoi region of the lattice \( \mathcal{L}_m \) is the convex polytope \( H_i \) defined as

\[
H_i := \{ x \in \mathbb{R}^m : \| x - y_i \|^2 \leq \| x - y_j \|^2, \text{ for all } j \}, \quad i \in \mathbb{Z},
\]

where \( \| x \| = \sqrt{x_1^2 + \ldots + x_m^2} \) and \( x = (x_1, \ldots, x_m) \in \mathbb{R}^m \). In (2.11), we let \( y_1 = (0, \ldots, 0) \), thus \( H_1 \) includes the origin \( y_1 \). Now \( G(L_m) \), the normalized second moment of \( H_i \) is defined as

\[
G(L_m) := \frac{1}{m} \int_{H_i} \| x - y_i \|^2 \frac{dx}{V(H_i)^{1/p}},
\]

where \( p := m/(m + 2) \) and \( V(H_i) := \int_{H_i} dx \) is the volume of \( H_i \) [24]. Note that all \( H_i, i \in \mathbb{Z} \), have the same shape. Thus, the normalized second moments and the volumes of \( H_i \) are all the same. Conway and Sloane have calculated the second moments of various lattices that yield values close to \( \inf_{L_m} G(L_m) \) for various dimensions, where the infimum is taken over all m-dimensional lattices [13], [14, Table 1]. For example, the hexagonal lattice, which is equivalent to the lattice \( A_2 \) in [13], is the optimal lattice in 2-dimensions. In the 3-dimensional case, the \( D_3^+ \) lattice (or equivalently the lattice \( A_3^+ \)) is a body-centered cubic lattice and optimal in 3-dimensions [4]. Furthermore, Conway and Sloane have found that that \( \inf_{L_3} G(L_2) = G(A_2) \cong 0.0802 \) and \( \inf_{L_3} G(L_3) = G(D_3^+) \cong 0.0785 \), and they have also conjectured a lower bound for \( \inf_{L_3} G(L_m) \) [15]. For the definitions of these lattices, see [13]. We now provide an asymptotic bound on the distortion of SAPQ over a uniform distribution input. This result will be used to derive an asymptotic bound (Theorem 1) over a more general distribution inputs.

**Lemma 1** Suppose that \( X \) is uniformly distributed over \( U := ([-a/2, a/2]^m) \), where \( a \) is a positive constant and \( m \) is a fixed positive integer. Then, for an increasing sequence \( (n, \ldots) \), such that \( n/2^n/m \rightarrow 0 \), where \( \alpha \) is a positive constant,

\[
\lim_{n \to \infty} \sup \left( \frac{n \alpha}{2^n} \right) \inf \mathcal{D}_{\text{SAPQ}} \leq G(L_m) \left[ \mu(U) \right]^{2/m}. \tag{2.13}
\]

Here, \( \mu \) is the lebesgue measure, and we assume that \( C_{i,j} \in \mathcal{L}_{n_\alpha} \).

**Proof of Lemma 1:** The derivation of Lemma 1 is shown in Appendix A.

Based on Lemma 1, we now extend the input source to an absolutely continuous distribution as follows.
Theorem 1 Suppose that $X$ has a joint density function $f$ with compact support and $f$ is bounded on $K^m$. Then,

$$\limsup_{n \to \infty} (n^{m/2})^{2/m} \inf D_{SAPQ} \leq G(L_m) ||f||_\rho, \quad (2.14)$$

where the functional $||f||_\rho$ is given by

$$||f||_\rho := \left[ \int f^\rho(x) dx \right]^{1/\rho}. \quad (2.15)$$

Proof of Theorem 1: The proof of Theorem 1 is given in Appendix B.

Corollary 1 Suppose that $E[||X||^{2+\epsilon}] < \infty$ for some $\epsilon > 0$ and $f$ is bounded on $R^k$. Then the inequality (2.14) holds.

Proof of Corollary 1: The proof of Corollary 1 is given in Appendix C.

From Theorem 1 or Corollary 1, we can obtain an asymptotic result

$$\limsup_{n \to \infty} (n^{m/2})^{2/m} \inf D_{SAPQ} \leq J_m ||f||_\rho, \quad (2.16)$$

where $J_m := \inf L_m G(L_m)$. It is clear from [41] that the optimal m-dimensional VQ is such that

$$\limsup_{n \to \infty} n^{2/k} \inf D_{VQ} \leq J_m ||f||_\rho \quad (2.17)$$

From [84] and [8], we know that the sequence on the left hand side converges. Further, from a well known (but yet unproven) conjecture, we know that the asymptotically optimal quantizer is a function of $J_m$ [24]. In other words,

$$\lim_{n \to \infty} n^{2/k} \inf D_{VQ} = J_m ||f||_\rho. \quad (2.18)$$

Therefore, if this conjecture were true (as is typically assumed), then SAPQ can achieve the asymptotically optimal m-dimensional VQ performance [52].

Based on the asymptotic result given by Theorem 1 and Corollary 1, we will discuss the achievable performance of SAPQ. As shown in (2.14) and (2.18), since SAPQ can achieve the performance $J_m ||f||_\rho$, the asymptotic performance of the optimal m-dimensional VQ, the advantages of SAPQ over PQ are the same as the VQ case over the scalar quantizer [57],[52]. We will now try to obtain more insight by studying the performance of SAPQ based on the two factors $G(L_m)$ (or $J_m$) and $||f||_\rho$.

B. Voronoi Region Shape: $G(L_m)$ (or $J_m$)

By observing (2.12) and (2.16), we can conclude that the factor $J_m$ is concerned with the shape of the Voronoi region of a quantizer. The gain achieved by this factor is called the space-filling advantage. Since $J_1 = 1/12$ and $\inf J_m = 1/2\pi\epsilon$, the achievable maximal gain through the shape of the Voronoi region is less than or equal to $10\log(J_1/\inf J_m) \cong 1.533$ dB.

In fact, in the literature, lattice VQs have been used to exploit this space filling advantage. Several important lattices can be described as the union of the cosets of a set [13]. Based on this fact, various encoding/decoding algorithms for lattice VQs have been proposed [14, (8)]. Such lattice VQs can be described by SAPQ. For example, the hexagonal lattice $A_2$, which yields the minimum $J_2 = G(A_2) \cong 0.0802$ in 2-dimensions, can be defined as

$$A_2 := \bigcup_{j=1}^2 \left\{ r_j + Z \times \{\cdots, -\sqrt{3}, -\frac{\sqrt{3}}{2}, 0, \frac{\sqrt{3}}{2}, \sqrt{3}, \cdots\} \right\}. \quad (2.19)$$
Here the coset representatives \( r_1 \) and \( r_2 \) are \( r_1 = (0, 0) \) and \( r_2 = (-1/2, \sqrt{3}/2) \). Since a coset of a product set is also a product set, \( A_2 \) is a union of two product sets. Hence, a truncated lattice of \( A_2 \) has the same structure as the SAPQ codebook in \( m \)-dimensions. Therefore a truncated lattice of \( A_2 \) can be implemented by SAPQ with \( \eta = 1 \), since we have two representatives.

An important lattice listed in [75] is the \( D^\perp_m \) lattice. For \( m \geq 2 \), \( D^\perp_m \) is the dual of the lattice \( D_m \) defined as

\[
D^\perp_m := \mathbb{Z}^m \cup \left( \left( \frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2} \right) + \mathbb{Z}^m \right). \tag{2.20}
\]

In a similar manner as in (2.19), it is clear that SAPQ can construct a truncated lattice of \( D^\perp_m \), for \( m = 1, 2, \ldots \), with only \( \eta = 1 \). Hence, SAPQ with \( \eta = 1 \) can construct the optimal lattice in 3-dimensions, since the \( D^\perp_3 \) lattice (or equivalently the lattice \( A^\perp_3 \)) is a body-centered cubic lattice and optimal in 3-dimensions [4]. The minimum value of \( G(D^\perp_m) \) is about 0.0747 at \( m = 9 \) [13]. Hence, the maximum gain is asymptotically \( 10 \log(1/G(D^\perp_9)) \approx 0.475 \) dB if we use lattice \( D^\perp_m \).

Another important type of lattice is the \( D_m \) lattice. For \( m \geq 2 \), \( D_m \) consists of the points \( x \in \mathbb{R}^m \) having integer coordinates with an even sum. The generator matrix \( U_{D_m} \) for lattice \( D_m \) is

\[
U_{D_m} := \frac{1}{2} \begin{pmatrix}
2 & 1 & 1 & \cdots & 1 \\
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1
\end{pmatrix}. \tag{2.21}
\]

Hence, \( D_m = \{ x \mid x = U_{D_m} p, p \in \mathbb{Z}^m \} \). Let \( D_{m,j} \) denote a subset of \( D_m \) defined as

\[
D_{m,j} := \{ x \mid x = U_{D_m} p', p' := \begin{pmatrix}
2m-2 & j_1 & \cdots & j_{m-1} \\
0 & 2p_2 & j_0 & \cdots & 2p_{m-1}
\end{pmatrix}, p_1, \ldots, p_{m-1} \in \mathbb{Z} \}, \tag{2.22}
\]

where \( j = j_{m-2}2^{m-2} + j_{m-3}2^{m-3} + \cdots + j_02^0 \) and \( j = 1, 2, \ldots, 2^{m-1} \). The lattice \( D_m \) can then be rewritten as

\[
D_m = \bigcup_{j=1}^{2^{m-1}} D_{m,j} = \bigcup_{j=1}^{2^{m-1}} \left[ U_{D_m}(0, j_0, j_1, \ldots, j_{m-2}) \right.
+ \{ x \mid x = U_{D_m}(p_1, 2p_2, 2p_3, \ldots, 2p_{m-1}), p_1, \ldots, p_{m-1} \in \mathbb{Z} \} \left. \right] \bigcup_{j=1}^{2^{m-1}} (r_j + \mathbb{Z}^m), \tag{2.23}
\]

where \( r_j := U_{D_m}(0, j_0, j_1, \ldots, j_{m-2}) \), for \( j = 1, 2, \ldots, 2^{m-1} \). Since from (2.23), we can represent \( D_m \) as the union of \( 2^{m-1} \) cosets of rectangular lattices, and since one rectangular lattice corresponds to a product codebook in SAPQ, we obtain \( 2^{m-1} = 2^\eta \). Hence, the side information required in this case is \( \eta = m - 1 \) which corresponds to the number of 1s in the diagonal of \( U_{D_m} \). This also means that \( D_m \) can be implemented by an \( m \)-SAPQ with \( \eta = m - 1 \).

In a similar way, we consider the lattice \( E_8 \). The lattice \( E_8 \) can be rewritten as \( E_8 = \{ x \mid x = U_{E_8} p, p \in \mathbb{Z}^8 \} \). Since \( 8 \) is the only even number that can be written as the sum of four squares, and since each square is odd, the lattice \( E_8 \) is formed by taking all possible sums of these squares. Hence, \( E_8 \) is a product codebook with \( \eta = 4 \). This means that \( E_8 \) can be implemented by a 4-SAPQ.
where \( p \) is written as a column vector and \( U_{E_8} \) is the generator matrix of \( E_8 \) given by

\[
U_{E_8} = \frac{1}{2} \begin{pmatrix}
2 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 2 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 2 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 2 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}.
\] (2.24)

Define \( E_{8,j} \) as the set

\[
E_{8,j} := \{ x \in U_{E_8} p' : p' := (p_0, p_1, p_2, p_3, 2p_4 + j_0, 2p_5 + j_1, 2p_6 + j_2, 2p_7 + j_3), p_0, \ldots, p_7 \in \mathbb{Z} \}.
\] (2.25)

Here \( j_0, \ldots, j_3 \in \{0, 1\} \) are given by \( j = 1 + j_0 2^0 + j_1 2^1 + j_2 2^2 + j_3 2^3 \), for \( j = 1, 2, \ldots, 2^4 \). Then

\[
E_8 = \bigcup_{j=1}^{16} E_{8,j}
\]

\[
= \bigcup_{j=1}^{16} \left[ U_{E_8}(0, 0, 0, 0, j_0, j_1, j_2, j_3) + \{x \in U_{E_8}(0, p_1, p_2, p_3, 2p_4, 2p_5, 2p_6, 2p_7), p_0, \ldots, p_7 \in \mathbb{Z} \} \right]
\]

\[
= \bigcup_{j=1}^{16} (r_j + \mathbb{Z}^8),
\] (2.26)

where \( r_j := U_{E_8}(0, 0, 0, 0, j_0, j_1, j_2, j_3) \), for \( j = 1, 2, \ldots, 16 \). Hence, a truncated lattice of \( E_8 \) can be implemented by the SAPQ with \( \eta = 4 \). Note that, in this case, the gain will be \( 10 \log(J_1/G(E_8)) \cong 0.654 \) dB.

We can construct examples of SAPQ for several other types of lattices, as described in [45].

**C. Distribution Shape: \( \|f\|_p \)**

The factor \( \|f\|_p \), which is defined in (2.15), is concerned with the joint density function \( f \). From this factor, we could potentially obtain a large gain in SAPQ based on the constrained-distortion quantizer [52]. From Holder's inequality [5], \( \|f\|_p \) is a nonincreasing sequence. Hence, depending on \( f \), we can expect some gain by increasing \( m \). Suppose that \( f \) is a uniform density function as

\[
f(x) = \begin{cases} 
\frac{1}{a^m}, & x \in \left(\left[-a/2, a/2\right] \cap \mathbb{Z}\right)^m \\
0, & \text{otherwise},
\end{cases}
\] (2.27)

where \( a \) is a positive constant. Then \( \|f\|_p = 12\sigma^2 \), where the variance \( \sigma^2 = a^2/12 \). Hence, for the uniformly distributed input case, we cannot expect any gain through this factor. However, suppose that \( f \) is a joint Gaussian density function given by

\[
f(x) = \frac{1}{(2\pi)^{m/2}(\det S)^{1/2}} \exp \left( -\frac{x^t S^{-1}x}{2} \right),
\] (2.28)

where \( S \) is the auto-covariance matrix of \( X \). Then

\[
\|f\|_p = 2\pi \rho^{-(m+2)/2}(\det S)^{1/m}.
\] (2.29)

It is well known that, if there is correlation between \( X_1, \ldots, X_m \), then we can reduce the distortion through the factor \( (\det S)^{1/m} \), since \( (\det S)^{1/m} \leq (\text{tr} S)/m \) [6]. The gain from this factor is known as the
memory advantage. The well known example that exploits this advantage is the discrete cosine transform. (Note that the term \((\det \mathbf{S})^{1/m}\) can be derived for other type of density function if we employ \(\| \cdot \|^2\) as a distortion measure \([43]\).)

Now focus on the factor \(\rho^{-(m+2)/2}\) in (2.29), which is dependent on the shape of \(f\). This factor \(\rho^{-(m+2)/2}\) is equal to \(3^{5/2} \approx 5.196\), for \(m = 1\), and monotonically decreases to \(e \approx 2.718\), as \(m \to \infty\). Hence, the achievable gain through the factor \(\rho^{-(m+2)/2}\) is \(10 \log (3^{5/2}/e) \approx 2.814\) dB. Furthermore, for the Laplacian density case, since

\[
\| f \|_p = 2 \rho^{-(m+2)} (\det \mathbf{S})^{1/m},
\]

the potential improvement is about 5.628 dB. The gain from the shape of the density function is called the shape advantage.

From the space-filling advantage and the shape advantage, even when the input \(X_1, \ldots, X_m\) is i.i.d. (or uncorrelated), we have the potential for an improvement of up to about \(4.347\) dB over PQ and up to about \(7.161\) dB over PQ for the Gaussian and the Laplacian density cases, respectively. Note that these maximum gains are the same as those obtainable from the corresponding theoretical bounds \([74]\).

2.4 Examples in Designing SAPQ

In this section, we provide several SAPQ design examples based on the theoretical observations made in Section 2.3 in order to more explicitly demonstrate the performance of SAPQ.

A. Uniform SAPQ Based on Lattices

In order to demonstrate the space-filling advantage gained from SAPQ, we will consider a uniformly distributed input as follows. Suppose that \(X_1, \ldots, X_m\) are i.i.d. and \(X\) has the uniform distribution in (2.27). As shown in Lemma 1, for this uniform density function, the gain comes only from the space-filling advantage. Hence, using the uniform density function, we can numerically observe the achievable gain from SAPQ by changing the shape of the Voronoi region.

If we use the same \(m\) Lloyd-Max scalar quantizers for the quantization of \(X\) based on the PQ in (2.2), then the Lloyd-Max quantizer is the uniform quantizer given by output points

\[
\frac{1}{2} (a + b) + b \cdot \ell, \text{ for } \ell = 1, 2, \ldots, n,
\]

where \(b = a/n\) is the step size. Hence, the average distortion in (2.2) can be rewritten as

\[
D_{\text{PQ}} = \sigma^2 2^{-2R},
\]

where \(\sigma^2 = a^2/12\) is the variance of \(X_i\) with the bit-rate \(R = \log_2 n\). Note that Shannon's lower bound (SLB) for the uniform density input is given by

\[
D_{\text{SLB}} = \frac{6}{\pi e} \sigma^2 2^{-2R},
\]

which is less than \(D_{\text{PQ}}\) of (2.32) by \(10 \log (\pi e/6) \approx 1.533\) dB. Hence, there is a potential of about 1.533 dB improvement, which is the same as the gain \(10 \log (J_1 / \inf J_m)\) from the space-filling advantage. For a finite dimension \(m\), VQ, tree-structured VQ, and TCQ can achieve a fraction of this potential improvement (i.e., better than \(D_{\text{PQ}}\)). It will be shown that SAPQ can also obtain this gain (without resorting to the complexities involved in the other schemes).

Now, for the uniform density function in (2.27), we provide several SAPQ design examples and numerical results as follows.
Example 2.1 (Lattice $D_m^n$): Let $C_j = \{y_{l,j}, \cdots, y_{n,j}\}$ (where $C_n$), for $j = 1, 2$, denote the codebooks of an SAPQ, where $n$ is a constant such that $\log n \in \mathbb{N}$. Let the codewords based on the lattice $D_m^n$ be defined as

$$y_{l,j} = -\frac{1}{2}(a + \frac{3b}{2}) + b \cdot \ell + b_j, \quad \text{for all } \ell, i, \text{ and } j,$$

where $b = a/n$, $b_1 = 0$, and $b_2 = b/2$. Note that this quantizer is SAPQ with $\eta = 1$. So the average distortion in (2.7) can be rewritten as

$$D_{\text{SAPQ}} = E \left\{ \min_{j \in \{1, 2\}} \frac{1}{m} \sum_{i=1}^{m} \min_{l \in \{1, \cdots, n\}} (x_i - y_{l,j})^2 \right\}$$

Here the bit-rate is $R = \log n + 1/m$. In Fig. 2.1, a block diagram of the SAPQ of this example is depicted. We have two different quantizers, $Q_1$ and $Q_2$ for $j = 1$ and $j = 2$, respectively. Each quantizer has two outputs; in Fig. 2.1, $Q_{\text{out}}(h)$ implies $m$ quantizer outputs, which is represented by a fixed length $\log n$ bits in input $h$, and $d_j(h) = (1/m) \sum_{i=1}^{m} \min_{l \in \{1, \cdots, n\}} (x_i(h) - y_{l,j})^2$. The comparator, COM, then compares the two values $d_1(h)$ and $d_2(h)$, and selects the index that has the lowest value. The output, $A_{\text{out}}(h)$, has 1 bit to indicate $Q_1$ or $Q_2$ for the multiplexer, MUX.

In Table 2.4, we compare the non-adaptive product quantizer with the adaptive quantizer of (2.34) for a uniform input with the variance $\sigma^2 = 1$. In this table, $D_{\text{PQ}}$ is calculated from the non-adaptive quantizer in (2.32). As shown in Table 2.4, at low bit-rates, $D_{\text{PQ}}$ is lower than $D_{\text{SAPQ}}$ due to the code-words of the sides of the uniform pdf. However, as the bit-rate increases, $D_{\text{SAPQ}}$ is lower than $D_{\text{PQ}}$ (as was discussed earlier, below the $D_m^n$ lattice in (2.20)). Note that we should expect; a gain (from our asymptotic analysis) to be $10 \log (J_1/G(D_8^n)) \approx 0.47$ dB, which appears to be consistenit with the results in Table 2.4.

Example 2.2 (Lattice $E_8$): Let $C_{i,j} = \{y_{i,1,j}, \cdots, y_{n,1,j}\}$ (where $C_n$), for $i = 1, \cdots, 8$ and $j = 1, \cdots, 16$, denote the codebooks of an SAPQ, where $n$ is a constant such that $\log n \in \mathbb{N}$. Let the codewords based on the lattice $E_8$ be defined as

$$y_{l,i,j} = -\frac{1}{2}(a + \frac{3b}{2}) + b \cdot \ell + b_{i,j}, \quad \text{for all } \ell, i, \text{ and } j,$$
where b = a/n, b_{i,j} := b r_{i,j}', and r_{i,j}' are given by the coset representatives r_j (\in \mathbb{R}^8) as follows. Let r_{i,j} be defined as

\[ r_j = U_{E_8}(0,0,0,0,j_0,j_1,j_2,j_3) \]
\[ := (r_{i,j}, \ldots, r_{8,j}), \tag{2.37} \]

where \( U_{E_8} \) is given in (2.24), \( j_0, \ldots, j_3 \in \{0,1\} \) and \( j := 1 + j_0 2^0 + j_1 2^1 + j_2 2^2 + j_3 2^3 \), for \( j = 1, 2, \ldots, 16 \) from (2.26). Then \( r'_j := (r'_{1,j}, \ldots, r'_{8,j}) \) is defined as

\[ r'_{i,j} := r_{i,j} - \lfloor r_{i,j} \rfloor, \tag{2.38} \]

where \( \lfloor c \rfloor, c \in \mathbb{R} \), is the largest integer less than or equal to c. Hence, the \( r'_j \) are given as follows.

\[
\begin{align*}
    r'_1 &= (0,0,0,0,0,0,0,0) & r'_2 &= 1/2(1,1,1,0,1,0,0,0) \\
    r'_3 &= 1/2(0,1,1,1,0,1,0,0) & r'_4 &= 1/2(1,0,0,1,1,1,0,0) \\
    r'_5 &= 1/2(0,0,0,1,1,0,1,0) & r'_6 &= 1/2(1,1,0,0,0,1,0,0) \\
    r'_7 &= 1/2(0,1,0,0,1,1,1,0) & r'_8 &= 1/2(1,0,1,0,0,1,0,1) \\
    r'_9 &= 1/2(1,1,1,1,1,1,1,1) & r'_{10} &= 1/2(0,0,0,1,0,1,1,1) \\
    r'_{11} &= 1/2(1,0,0,0,1,0,1,1) & r'_{12} &= 1/2(0,1,1,0,1,0,1,1) \\
    r'_{13} &= 1/2(1,1,0,0,0,1,0,1) & r'_{14} &= 1/2(0,0,1,0,1,1,0,1) \\
    r'_{15} &= 1/2(1,0,1,1,0,0,0,1) & r'_{16} &= 1/2(0,1,0,1,1,0,0,1)
\end{align*}
\]

Therefore, this quantizer is the \( \text{SAPQ} \) with \( \eta = 4 \) and the bit-rate is \( R = (\log n + 0.5) \). (Note that the quantizer in Example 2.1 has only \( r'_1 \) and \( r'_2 \).) The \( \text{SAPQ} \) distortion of this example is given by

\[
D_{\text{SAPQ}} = E \left\{ \min_{j \in \{1, \ldots, 16\}} \min_{t \in \{1, \ldots, n\}} \sum_{i=1}^{n} (X_i - y_{t,i,j})^2 \right\} \tag{2.40} \]

The results are summarized in Table 2.4. Note that from our asymptotic analysis, we would expect a gain of approximately 0.65 dB on lattice \( E_8 \), which is consistent with our results in Table 2.4. In a similar manner, we can design an \( \text{SAPQ} \) based on the lattice \( E_7 \) with \( \eta = 3 \).

B. Nonuniform SAPQ

We now introduce several examples to demonstrate the gain from the space-filling and shape-advantages for non-uniform sources.
The design problem of SAPQ is to find an optimal codebook that achieves the distortion \( D_{\text{SAPQ}} \) for a fixed rate \( R \). However, finding such an optimal codebook is not easy for the non-uniformly distributed inputs. In order to find (sub)optimal codebooks, we have developed a clustering algorithm that uses a large number of samples as a training sequence (TS) for given values of \( m \), \( n_i \), and \( q \). But this TS size is still substantially less than that of traditional VQ or modified schemes. Let \( x_{1,\ell}, \ldots, x_{m,\ell} \) denote the \( \ell \)th training sample in a given TS that has \( M \) samples, where a sample has \( m \) training points.

The first part of our algorithm quantizes \( m \) training points in each sample using \( 2^n \) different codebooks and then selects a codebook that yields the minimal distance (given in (2.6)) for the sample. The second part of the algorithm updates the codebooks using the partitioned TS in the quantization process of the first part. These two parts are then iteratively applied to the given TS. The clustering algorithm is described below.

**Clustering Algorithm (SAPQ)**

0. Initialization (\( k = 0 \)): Given codebook sizes \( n_i, i = 1, \ldots, m \), sample size \( m \), side bits \( q \), distortion threshold \( \epsilon \geq 0 \), initial codebook \( C_0 \), and TS \( \{ (x_{1,\ell}, \ldots, x_{m,\ell}) \}_{\ell=1}^{M} \), set \( D_{-1} = \infty \).

1. Given codebook \( C_k = \bigcup_{j=1}^{2^n} (C_{1,j} \times \cdots \times C_{m,j}) \), where \( C_{i,j} \subseteq C_{n_i} \), find \( 2^n \sum_{i=1}^{m} n_i \) partitions of each training points in the TS for the corresponding \( 2^n \sum_{i=1}^{m} n_i \) codewords, where each training point's codeword is determined by the following quantization

\[
d_{\ell} := \min_{j} \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i,j}} (x_{j,\ell} - y)^2, \quad \ell = 1, \ldots, M
\]  

Next, we compute the average distortion \( D_k \), for the \( k \)th iteration, given by

\[
D_k := \frac{1}{M} \sum_{\ell=1}^{M} d_{\ell}.
\]  

2. If \( (D_{k-1} - D_k)/D_k \leq \epsilon \), stop. \( C_k \) is the final codebook. Otherwise continue.

3. Increase \( k \) by 1. Compute centroids for each of the \( 2^n \sum_{i=1}^{m} n_i \) partitions and replace the codewords in \( C_{i,j} \) by the new \( 2^n \sum_{i=1}^{m} n_i \) centroids. Go to Step 1.

It can be shown using similar techniques as in the case of the Lloyd-Max algorithm or the K-means algorithm [2] that \( D_k \) is a decreasing sequence. Thus, \( D_k \) converges to a (local) minimum, which depends
on the initial codebook $C_0$. The next example shows an effect of the initial codebook in the clustering algorithm.

![Figure 2.2: The codebooks of SAPQ in m-dimensions for different initial guess (Gaussian i.i.d. input with the variance $1$, $n = 2$, $m = 2$, and $\eta = 1$. Note that each codebook is the union of two product codebooks.). (a) Initial Guess 1 (distortion: -6.93 dB). (b) Initial Guess 2 (distortion: -6.14 dB).](image)

**Example 2.3 (Initial Guess in Clustering Algorithm):** The clustering algorithm can be used to effectively design the SAPQ codebook using the TS that has an underlying distribution function. However, the performance of the designed SAPQ is very dependent on choosing the initial codebook $C_0$. An example of the different choices of the initial guess is illustrated in Fig. 2.2 by plotting the codebook of the SAPQ in m-dimensions. (Note that, in Fig. 2.2, each figure has a codebook $C$ that is the union of two product codebooks ($2^2 = 2$), each product codebook has 4 codewords ($n^m = 4$), and $C_{1,j} = C_{2,j}$.) Figs. 2.2(a) and (b) show the converged codebooks in the clustering algorithm. However, the corresponding initial codebooks also have similar arrangements to the converged codebooks. This fact implies that the designed SAPQ codebook is very dependent on the initial codebook $C_0$. Furthermore, if $n$ and $\eta$ are large, then we have many choices of the initial codebooks. Hence, finding a globally optimal codebook $C$ for an input is quite difficult except for several trivial cases. In Fig. 2.2(a), we have employed a simple split method, to determine the codebook $C$. The split method doubles the number of the product codebooks by adding and subtracting a small constant $\varepsilon$. For the generation of an initial codebook $C_0$ from the split method, we need a start codebook that is denoted by $C_0^0$ in $\mathbb{R}^2$. The start codebook $C_0^0 = (C_{1,1}^0 \times C_{m,1}^0)$ contains codebooks $C_{i,1}^0$ that belong to $C_{n,1}^0$, where $C_{i,1}$ is the Lloyd-Max quantizer that is optimal for $X_i$.

**Initial Codebook Guess (Split Method for SAPQ)**

0. Initialization ($k = 0$): Given codebook size $n_i^0$, sample size $m$, side bits $\eta$, split constant $\varepsilon \geq 0$, start codebook $C_0^0 \subset \mathbb{R}^m$, and TS \((z_1, \ldots, z_m, t)\)_{t=1}^M.

1. If $k \geq \eta$, stop. $C_k^0$ is the initial codebook $C_0$ for the clustering algorithm. Otherwise continue.

2. Increase $k$ by 1. Construct a new codebook $C_k^0 = \bigcup_{j=1}^{2^k} (C_{1,j}^{k-1} \times \cdots x C_{m,j}^{k-1})$, by doubling the number of codebooks from $C_0^{k-1} = \bigcup_{j=1}^{2^k-1} (C_{1,j}^{k-1} x \cdots x C_{m,j}^{k-1})$ as follows.

$$C_{i,j}^{k} = -\varepsilon + C_{i,j}^{k-1} \quad \text{and} \quad C_{i,d^k+j}^{k} = \varepsilon + C_{i,j}^{k-1},$$

(2.43)
3. Given $C_{0,i}^k$, find $2^k \sum_{i=1}^{m} n_i^j$ partitions of training points according the quantization

$$
\min_{j \in \{1,2,\ldots,2^k\}} \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i,j}^k} (x_i^\ell - y)^2, \text{ for } \ell = 1, \ldots, M
$$

(2.44)

Compute the centroids for each of the $2^k \sum_{i=1}^{m} n_i^j$ partitions and replace the codewords in $C_{i,j}^k$ by the new $2^k \sum_{i=1}^{m} n_i^j$ centroids. Go to Step 1.

Figure 2.3: The codebooks of SAPQ in $m$-dimensions for different initial guesses (Gaussian Markov-1 source with the variance 1 and the correlation coefficient 0.9, $n = 2$, $m = 2$, and $\eta = 1$). (a) Split method with $\varepsilon = 0.01$ (distortion: -8.07 dB). (b) Split method with $\varepsilon = 1$ (distortion: -8.73 dB).

In this split method, we set $\varepsilon = 0.001$ during the simulation. Fig. 2.3 illustrates an example of the constant $\varepsilon$ in the split method for a correlated input.

**Example 2.4 (A Codebook-Constrained SAPQ):** Note that the SAPQ in (2.7) requires at most $m2^n$ different codebooks. Hence, if $m$ is large, the decoder needs a large memory for the codebooks and the codebook design complexity may be high. In order to reduce the required number of codebooks, one possibility is to use the same codebooks in calculating the distance of (2.6) under an assumption that the random variables $X_1, \ldots, X_m$ are identically distributed. In other words, $C_{i,j}$ are set equal for $i = 1, \ldots, m$. We can regard this scheme as a codebook-constrained SAPQ and in this case the average distortion is given as

$$
D_{\text{SAPQ}} = E \left\{ \min_{i \in \{1,2,\ldots,2^k\}} \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i,j}^k} (X_i - y)^2 \right\}
$$

(2.45)

Here the index $i$ is omitted in the codebook notation. Note that the number of required codebooks is reduced to $2^J$ and the bit-rate is given by

$$
R = \log_2 n + \frac{\eta}{m},
$$

(2.46)
if $C_j \in C$, for all $j$. As shown in the asymptotic analysis of Section 2.3, increasing $m$ for a fixed value of $n$ yields more gain over PQ in the SAPQ of (2.7). However, for the codebook-constrained SAPQ case, the decrease in distortion can be seen to diminish for large values of $m$, and the distortion will eventually increase and converge to that of the $n$-level quantizer [45]. Therefore, to obtain gains in the codebook-constrained SAPQ, it is important to use as large a value for $m$ (and $n$) as possible, while keeping the ratio $m/n$ small (note that since increasing $n$ increases the total bit-rate, this implies that for a given bit-rate the side information $\eta$ should be accordingly decreased). Furthermore, if we employ the split method as an initial guess in the clustering algorithm, the distortion of the codebook-constrained SAPQ is nearly the same as that of SAPQ in (2.7), especially for $n \geq 4$ as shown in Fig. 2.4. In fact, we have found through extensive simulation studies that for fixed values of ratio $m/n$ and bit-rate, increasing $n$ results in each of the $m$ codebooks of a codebook sequence in SAPQ to approach a single codebook (i.e., become equal to one another) [45]. Therefore, for a relatively large $n$ (compared to $\eta$) and a fixed ratio of $m/n$, it is advantageous to use the codebook-constrained SAPQ, since its performance will closely approximate that of SAPQ and the number of required codebooks is $2^m$. Using these design guidelines allows us to reduce the memory requirement by a factor of $m$ without a major compromise on performance. Note that the required size of the TS in the clustering algorithm is proportional to the number of the total codewords [41]. Hence, the required TS size and clustering time are also reduced by factor $m$. In the next simulation, we will show a result using the above design guidelines.

**Example 2.5 (Codebook-Constrained SAPQ and PQ):** In Fig. 2.5, the codebook-constrained SAPQ in Example 2.4 and the PQ in (2.2) are compared for the Gaussian and Laplacian i.i.d. sources. The quantizers in PQ are the Lloyd-Max quantizers [56],[65]. For example, for $m = 16$ and $\eta = 4$ in Fig. 2.5(a), the distortion of the codebook-constrained SAPQ shows a 1.8 dB improvement over PQ. This is consistent with our discussion in the previous section regarding the Laplacian source yielding more gains. As shown in Fig. 2.5(b), the distortion of the codebook-constrained SAPQ shows a 2.9 dB improvement over PQ at $m = 16$ and $\eta = 4$.

If we implement a traditional VQ at one of the bit-rates shown in Fig. 2.5, for example $R = 4.25$ bits/point, with vector dimension $m = 16$, the number of codewords required is $2^m R \approx 2.951 \times 10^{20}$. Hence we would need a very large memory to store $2.951 \times 10^{20}$ vectors (or $4.722 \times 10^{21}$ points in $\mathbb{R}$, since each codeword in VQ belongs to $\mathbb{R}^m$). Note that since the memory requirement is critical in implementing both the encoder and decoder, it is quite impractical to implement VQ at this bit-rate. Further, for a good codebook design for VQ using TS, we need more than $1.476 \times 10^{24}$ training vectors; where we set the

![Figure 2.4: Distortions (dB) of SAPQ and the codebook-constrained SAPQ for different values of $\eta$. (Gaussian i.i.d. input with the variance 1 and the ratio $m/n = 1$.)](image-url)
training ratio (TS size to codebook size) to 5,000, to ensure a good codebook design for the underlying distribution [41]. Hence, it is also nearly impossible to design such a VQ, now from the point of view of codebook design, for the bit-rates shown in Fig. 2.5. On the other hand, the number of the codewords in SAPQ at the same bit-rate and vector dimension is $2^n = 256$, which is a reasonable number for a practical implementation of the quantizer. Hence we need a small memory for 16 vectors (or 256 points). Also, for designing a good codebook for SAPQ, we need only $8 \times 10^4$ training vectors assuming the same training ratio of 5000. (Note that the codewords of SAPQ belong to R.) As shown in (2.10) even though the minimal m-dimensional VQ distortion is less than that of the minimal SAPQ distortion having the sample size m, we cannot use the m-dimensional VQ because of its high encoding and designing complexity. On the other hand, by using the proposed SAPQ, we can obtain VQ-level performance at such high bit-rates.

![Graph](image)

Figure 2.5: The codebook-constrained SAPQ and PQ. $n = 16$ and the results are obtained by varying m (m = 8, 16, and 24), for each $\eta$. (a) Gaussian i.i.d. with the variance of 1. (b) Laplacian i.i.d. with the variance of 1.

Furthermore, since SAPQ has a scalar quantizer structure, the codewords of SAPQ belong to $\mathbb{R}$, and hence SAPQ can quantize an error signal from the scalar value predictor in a DPCM: coder. However, VQ requires codewords that belong to $\mathbb{R}^m$. Hence, employing the scalar value predictor in a VQ (even modified VQ) is inefficient. An application of SAPQ for the differential PCM (DPCM) of image signals will be introduced in Section 5.

C. Entropy Constrained SAPQ

If we do not use the entropy coder for the quantized output, then we can obtain a large gain using SAPQ, as shown in Example 2.5. However, if we employ the entropy coder, then we still achieve a non-trivial gain from SAPQ, albeit, not as large. The next example shows an entropy-constrained SAPQ. Consider a mid-tread uniform quantizer in $\mathbb{R}$ with codebook given by

$$C_U := \{0, \pm b, \pm 2b, \cdots\}.$$  \hspace{1cm} (2.47)

Here $b (> 0)$ is the step size. Let $H_U$, the entropy of the quantizer, be defined as

$$H_U := \sum_{\ell=1}^L P_\ell \log_2 P_\ell,$$  \hspace{1cm} (2.48)
where \( P_t \) is the non-zero probability that the quantizer output is the \( \ell \)th codeword in \( C_U \) and suppose that \( P_t \neq 0 \) for \( \ell = 1, \ldots, L \), and zero otherwise. In the case of high entropies of \( H_U \), the PQ that employs the uniform quantizer will yield the minimum entropy, and that this minimum is higher than the rate distortion bound by only about \( \frac{1}{4} \) bit, which corresponds to about 1.533 dB. The next example will show that using SAPQ can reduce this 1.533 dB gap.

Example 2.6 (Entropy-Constrained SAPQ based on lattice \( D^L_m \))

Suppose that \( r_i = (b/40) \cdot i (\in \mathbb{R}) \), for \( i = 0, \ldots, 20 \). For a given \( i \), consider two cosets, \( C_1 = r_i + C_U \) and \( C_2 = -r_i + C_U \) as the codebooks in an SAPQ with \( \eta = 1 \). Note that, if \( i = 0 \), then \( C_1 \) is a mid-rise codebook and \( C_1 = C_2 \); if \( i = 20 \), then \( C_1 \) is a mid-rise codebook and \( C_1 \neq C_2 \), and if \( i = 10 \), then the set \( C = C_1 \times C_2 \) is a lattice that is equivalent to \( D^L_m \). For the \( i \geq 21 \) case, the set \( C \) is equal to one of the cases for \( i = 0, \ldots, 20 \). Hence, we will consider only the cases for \( i = 0, \ldots, 20 \), i.e., \( 0 \leq r_i \leq b/2 \). Let \( H_{SAPQ} \) denote an entropy of the SAPQ using \( C \) as a codebook in \( m \)-dimensions, where \( H_{SAPQ} \) is defined as

\[
H_{SAPQ} := -\Pr\{C_1 \text{ used for } X\} \sum_{\ell=1}^{L_1} P_{1,\ell} \log_2 P_{1,\ell} + \Pr\{C_2 \text{ used for } X\} \cdot \sum_{\ell=1}^{L_2} P_{2,\ell} \log_2 P_{2,\ell} + \frac{1}{m}. \quad (2.49)
\]

Here \( 1/m \) is the side information given by the SAPQ with \( \eta = 1 \), and it is assumed that \( P_{1,\ell} \neq 0 \), for \( \ell = 1, \ldots, L_1 \), and zero otherwise. Several numerical results are summarized in Table 2.4 and Table 2.4 for Gaussian and Laplacian density functions, respectively. In these tables, the step size of the SAPQ codebook \( C \) is denoted by \( b_{SAPQ} \) and the bit-rates \( R \) of the PQ and SAPQ are given by (2.3) and (2.8), respectively. Note that the rates of both the PQ and SAPQ satisfy \( H_U \leq R \) and \( H_{SAPQ} \leq R \), respectively. In this simulation, we conducted the SAPQ for the 21 values of \( r_i \) and found \( r_i \) that yields the minimum distortion. (Variation of \( H_{SAPQ} \) is very small for the various values of \( r_i \), especially for the high entropy case.) For the Gaussian density function case (Table 2.4), most of the results show the minimum distortion at \( r_{10} = b_{SAPQ}/4 \). Since \( D_{SLB} = \sigma^2/2^R \), where the variance \( \sigma^2 = 1 \), the distortion of PQ at \( H_U = 4.170 \) has about 1.53 dB difference from the SLB as shown in Table 2.4. In this case, the SAPQ achieves about 0.49 dB gain over the PQ as expected from the discussion following (2.20). However, this gain decreases as the entropy \( H_{SAPQ} \) decreases. This fact can be explained in the similar manner to Example 2.1. Table 2.4 shows the numerical result for a Laplacian density function. At \( H_U = 4.073 \), the difference between \( D_{PQ} \) and the SLB is about 1.56 dB and the gain from the SAPQ is about 0.46 dB. (Note that \( D_{SLB} = \sigma^2/2^{1.5} \).) For the Laplacian case, we again note that for large step sizes, \( r_i \) is less than \( r_{10} = b_{SAPQ}/4 \) and the performance of SAPQ can even be worse than that of the PQ. However, in both the cases, if the step size is smaller than or equal

<table>
<thead>
<tr>
<th>Step Size: ( b_{SAPQ} )</th>
<th>( H_{SAPQ} (D_{PQ}) )</th>
<th>Distortion (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>4.170</td>
<td>-23.58 (-24.07)</td>
</tr>
<tr>
<td>0.5</td>
<td>3.179</td>
<td>-17.56 (-18.04)</td>
</tr>
<tr>
<td>1</td>
<td>2.213</td>
<td>-11.55 (-12.01)</td>
</tr>
<tr>
<td>2</td>
<td>1.331</td>
<td>-5.44  (-6.01)</td>
</tr>
<tr>
<td>4</td>
<td>0.458</td>
<td>-1.06  (-1.14)</td>
</tr>
</tbody>
</table>
Table 2.4: Entropy-CONSTRAINED SAPQ Based on $D_m^\perp$ (Example 2.6)

<table>
<thead>
<tr>
<th>Step Size: $\delta_{\text{SAPQ}}$</th>
<th>$H_{\text{SAPQ}} (H_U)$</th>
<th>Distortion (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>4.073</td>
<td>-23.59</td>
</tr>
<tr>
<td>0.5</td>
<td>3.088</td>
<td>-17.62</td>
</tr>
<tr>
<td>1</td>
<td>2.134</td>
<td>-11.77</td>
</tr>
<tr>
<td>2</td>
<td>1.255</td>
<td>-6.43</td>
</tr>
<tr>
<td>4</td>
<td>0.509</td>
<td>-2.38</td>
</tr>
</tbody>
</table>

...to the standard deviation (in these cases, 1), then as shown in Tables 2.4 and 2.4, there is a reasonable gain. Note that these gains come from the space-filling advantage.

Suppose that an entropy coding scheme is employed in the SAPQ of Example 2.6 and denote the resultant bit-rate by $\tilde{R}$. The optimal entropy coding, where $\tilde{R} = H_{\text{SAPQ}}$, can only be reached if the probability $P_{j,e}$ satisfies the Shannon-Fano integral constraint. Otherwise, the bit-rate $\tilde{R}$ that results from entropy coding will be slightly higher than $H_{\text{SAPQ}}$. It is useful to use entropy coding on vector (rather than on single outputs) in order to reduce the difference between $\tilde{R}$ and $H_{\text{SAPQ}}$. For example, a three-dimensional variable length coding scheme is employed for the DCT coefficient coding in ITU-T, H.263 [38].

We can also use a non-uniform quantizer with entropy coding in order to obtain a higher gain than does the uniform quantizer with entropy coding. An application of SAPQ to the quantizers for the very low bit-rate video coding scheme based on H.263 is studied in [46].

2.5 Conclusion

In this section, to reduce the encoding complexity usually required in vector quantizers, we have proposed a structurally constrained VQ. The proposed quantizer is a unique feed-forward adaptive quantizer having a short adaptation period, and is hence called the sample-adaptive product quantizer (SAPQ). SAPQ allows us to obtain VQ-level performance, especially for high bit-rates. Unlike the modified VQ techniques, such as the tree-structured VQ and classified VQ, the proposed quantization scheme is based on a scalar quantizer structure, and is hence very appealing from a practical implementation point of view. Further, SAPQ can even be applied for high bit-rates, where conventional VQ (or even modified VQ) techniques are very difficult to use. The scalar quantizer structure of SAPQ also allows us to easily apply it to current coding systems, and generate VQ-level performance.
3. DCT Coefficient Quantization Using Non-Zero Level Codebooks based on the Sample-Adaptive Product Quantization

3.1 Introduction

Quantization is the elementary technique in reducing the amount of data while possibly introducing a small error, called quantizer distortion. When we consider the memoryless and instantaneous quantizer, the quantizer can be classified by the following two classification rules; the types in the first class are the uniform and nonuniform quantizer types and the types in the second class are the mid-tread and mid-rise quantizer types [39]. Among the quantizer types, the mid-tread type has the zero level in the output levels, and the mid-rise type does not have the zero level. The standard of the still image coding system, ISO (International Standard Organization)/IEC (International Electrotechnical Committee) JPEG (Joint Picture Experts Group), is based on the mid-tread uniform quantizers that has the zero level [82]. Especially at smooth areas in an image, the AC coefficients of the discrete cosine transform (DCT) are nearly zero. Hence, the quantizer should have the zero level to prevent the granular noise and obtain a low quantizer distortion. In the video coding standards, such as ISO/IEC MPEG (Moving Pictures Experts Group)-1/MPEG-2 and ITU-T (International Telecommunication Union – Telecommunication Standardization Sector) H.261/H.263, which are based on a hybrid of the block-based DCT and the motion compensation technique, we have the same notion of employing the mid-tread uniform quantizer. In this case, the zero level is required for the smooth areas or well motion compensated areas. In order to increase the gain, the image/video coding systems employ a multidimensional entropy coder by combining VLC, the run-length coding, etc, to circumvent the Shannon-Fano integral constraint. Since the gain from the run-length coding is achieved when the quantizer output sequence has many sequential zeros, a quantizer that has a wide quantizer region for the zero level is used to reduce the bit rate [38]. We call this quantizer the mid-tread uniform quantizer with dead zone. For example, the distance between the zero level and the next level is 50% wider than the others in the H.263 quantizer for the INTER blocks.

To obtain more gains especially over the quantizer part of the current standards, it is necessary to use the vector quantizer (VQ) instead of the scalar quantizer, i.e., VQ can achieve a gain which cannot be achievable through the scalar quantizer. However, the major disadvantage of VQ is the encoding complexity which increases dramatically as the vector dimension or bit rate increases. The current video coding standard systems are based on high bit-rate quantizers, even when the coding systems are designed for a low bit-rate transmission environment. Hence, applying VQ to the current video coding structure is very difficult. In order to overcome this problem, various modified VQ techniques have been proposed [25], e.g., tree-structured VQ, classified VQ, lattice VQ, and the trellis coded quantizer (TCQ) [25],[80],[13],[44]. However, since such schemes are still based on the VQ structure, the application areas of these schemes are relatively limited. TCQ is gaining interest for their performance with a moderate encoding complexity [25],[55]. However, since TCQ requires the special techniques: the trellis encoder and the Viterbi decoder, implementing TCQ is complex.

In this section, we propose an adaptive quantizer scheme (AQS) by employing a feed-forward adaptive quantizer, SAPQ, for the INTER block quantization of the ITU-T H.263 version 2, called H.263+. Since the structure of this quantizer is based on scalar quantizers, we can readily apply SAPQ to the current quantizer schemes of the video standards. What is remarkable about SAPQ is that it can be used to obtain VQ-level performance with scalar quantization-level complexity.

This section is organized as follows. In Section 3.2, we concisely describe the uniform SAPQ. We introduce the proposed AQS for H.263+, the low bit-rate video coding, in Section 3.3. In Section 3.4, the design of SAPQ to the INTER block quantization in H.263+ is shown, and simulation results with discussions are given. We then summarize and conclude the section in the last section.
3.2 Uniform Sample-Adaptive Product Quantization

In this section, we will study the uniform SAPQs based on several lattices in preparation for the design of the SAPQ codebook for the proposed AQS. If we employ the entropy coder for the outputs of a quantizer, then it is known that using a uniform VQ, such as the lattice VQ, can achieve the performance that is very close to Shannon’s bound. Note that, in this section, the SAPQ that is to be employed in the proposed AQS already has the fundamental idea of the lattice VQ, i.e., exploiting space-filling advantage. Hence, we now demonstrate several SAPQs that can implement several important lattice VQs. This demonstration will be a guideline to design the SAPQ codebooks in the following sections.

In order to study the performance of the uniform SAPQs based on lattices, suppose that \( X_1, \ldots, X_m \) are i.i.d. and \( X_i \) has the uniform distribution in (2.27). Note that, for this uniform density function, the gain comes only from the space-filling advantage. Hence, using a synthetic data that is uniformly distributed, we can numerically observe the achievable gain from the SAPQ by changing the shape of the Voronoi region.

Let \( C_{i,j} := \{y_{i,j}, \ldots, y_{m,i,j}\} (\in \mathcal{C}_n) \), for \( i = 1, \ldots, m \) and \( j = 1, \ldots, 2^n \), denote the codebooks of an SAPQ (see (2.7) for the SAPQ codebooks), where \( n \) is a constant such that \( \log n \leq N \). Let the codewords be defined as

\[
y_{i,j} := -\frac{1}{3}(a + \frac{3b}{2}) + b \cdot \ell + b_{i,j}, \quad \text{for all } \ell, i, \text{ and } j,
\]

where \( b = a/n, b_{i,j} := r_{i,j}' \), and \( r_{i,j}' \) are given by the coset representatives (CRs) \( r_{i,j}' := (r_{i,j}', \ldots, r_{m,j}') (\in \mathbb{R}^m) \) of a lattice. Using this notation, the distortion in (2.7) can be rewritten as

\[
D_{SAPQ} = E\left\{ \min_{j \in \{1, \ldots, 2^n\}} \frac{1}{m} \sum_{i=1}^{m} \min_{\ell \in \{1, \ldots, n\}} (X_i - y_{\ell,i,j})^2 \right\}.
\]

From Example 2.1, for the implementation of an SAPQ based on the lattice \( D_{m}^{+} \), the CRs are given by

\[
\text{CR-}D_{m}^{+}: \quad r_{i}' = (0, \ldots, 0) \quad \text{and} \quad r_{i}' = 1/2(1, \ldots, 1),
\]

for integers \( m \geq 2 \), and \( \eta = 1 \). (Note that the side bit rate is \( \eta/m = 2/m \), and the SAPQ has two different codebooks.) A block diagram of this quantizer is illustrated in Fig. 2.1. We now observe the performance of the uniform SAPQ that is designed based on lattice \( D_{m}^{+} \) in (3.3) for different \( m \). It is known that, for the fixed \( \eta = 1 \) case, the maximum gain at \( m = 9 \) is about 0.475 dB over the scalar quantizer in (2.32) [15]. Further, as \( m \) increases above 9, the gain is reduced. In order to verify this fact by experiment, numerical results are illustrated in Fig. 3.1 for the uniform pdf of (2.27) with variance 1. In this figure, the gain is \( R = 8 + 1/m \). However, we can notice a difference in the numerical results in Fig. 3.1. As \( m \) increases from 2, the gain increases and we have a 0.45 dB gain at \( m = 8 \). However, at \( m = 9 \), the gain is slightly reduced to 0.43 dB, and then at \( m = 10 \), achieves the maximum gain 0.46 dB. As \( m \) increases from 10, the gains are monotonically reduced. For example, the gain is 0.39 dB at \( m = 32 \).

For \( m = 8 \), the gain from the codebooks of (3.3) is about 0.475 dB. However, if we set \( \eta = 4 \) and use the lattice \( E_8 \), then we can obtain more gain. For the implementation of the lattice \( E_8 \), the CRs are given by

\[
\text{CR-}E_8: \quad r_{i}' = \{0, 0, 0, 0, 0, 0, 0, 0\}, \quad r_{i}'' = 1/2(1,1,1,0,1,0,0,0),
\]

(3.4)

\[
r_{3}' = 1/2(0,1,1,1,0,1,0,0), \quad r_{4}' = 1/2(1,0,0,1,1,1,0,0),
\]

\[
r_{5}' = 1/2(0,0,1,1,1,0,1,0), \quad r_{6}' = 1/2(1,0,1,0,0,1,0),
\]

\[
r_{7}' = 1/2(0,1,0,0,1,1,1,0), \quad r_{8}' = 1/2(1,0,1,0,0,1,1),
\]

\[
r_{9}' = 1/2(1,1,1,1,1,1,1,1), \quad r_{10}' = 1/2(0,0,0,1,0,1,1,1),
\]

\[
r_{11}' = 1/2(1,0,0,0,1,0,1), \quad r_{12}' = 1/2(0,1,1,0,0,0,1,1),
\]

\[
r_{13}' = 1/2(1,1,0,0,0,1,0,1), \quad r_{14}' = 1/2(0,0,1,0,0,1,1),
\]

\[
r_{15}' = 1/2(1,0,0,1,0,1,0,0), \quad r_{16}' = 1/2(0,1,0,1,1,0,0,1),
\]

\[

\text{The gain in dB is defined as } 10\log[D_{PQ}(R)/D_{SAPQ}(R)].
and $\eta = 4$, where $m = 8$ (Example 2.2). (Note that the side bit rate is $\eta/m = 0.5$.) The expected gain from the quantizer using CRs given by (3.4) is approximately 0.654 dB. (A numerical result shows 0.65 dB gain at $n = 256$.) For $m = 8$, when $\eta = 1$, we can design the codebooks of SAPQ based on the lattice $D_m^\perp$ as in (3.3), and when $\eta = 4$, we can design the codebooks based on the lattice $E_8$ as in (3.4).

Based on these two lattices, we now design the SAPQ codebooks for $\eta = 2$ and 3, and illustrate the numerical results. For $\eta = 2$, we can choose 4 different CRs from (3.4). The bit rate is $R = 8.25$. We consider following two examples in choosing four CRs,

$$
\text{CR-1: } r'_1 = (0,0,0,0,0,0,0,0), \quad r'_2 = 1/2(1,1,1,1,1,1,1), \quad r'_3 = 1/2(1,0,0,1,1,0,0), \quad r'_4 = 1/2(0,1,1,0,0,0,1,1)
$$

and

$$
\text{CR-2: } r'_1 = (0,0,0,0,0,0,0,0), \quad r'_2 = 1/2(0,1,1,1,0,1,0,0), \quad r'_3 = 1/2(0,0,1,1,1,0,1,0), \quad r'_4 = 1/2(0,1,0,0,1,1,1,0)
$$

However, the two examples show different gains, which are calculated by experiment. CR-1 shows 0.36 dB gain, but CR-2 only shows 0.16 dB gain. We can consider different combinations of the CRs outside the CRs of (3.4). However, it seems that the best one is CR-1 for $\eta = 2$. (Note that the gain is less than that of the $\eta = 1$ case of (3.3). This implies that increasing the side bit rate does not always guarantee a better gain.) In a similar manner, we study the $\eta = 3$ case ($R = 8.375$), where we have 8 different CRs. From the CRs in (3.4), consider the following 8 CRs,

$$
\text{CR-3: } r'_1 = (0,0,0,0,0,0,0,0), \quad r'_2 = 1/2(1,1,1,1,1,1,1,1), \quad r'_3 = 1/2(1,1,1,0,1,0,0,0), \quad r'_4 = 1/2(0,0,0,1,0,1,1,1), \quad r'_5 = 1/2(0,1,1,1,0,1,0,0), \quad r'_6 = 1/2(1,0,0,0,1,0,1,1), \quad r'_7 = 1/2(1,0,0,1,1,0,1,0), \quad r'_8 = 1/2(0,1,1,1,0,0,1,1)
$$

The SAPQ using CR-3 produces 0.40 dB gain. However, if we consider $r'_1, r'_2, \ldots, r'_8$ of CR-E_8 for the 8 CRs, then the gain is 0.34 dB. It seems that the $\eta = 3$ case also yields a lesser gain than the $\eta = 1$ case, even though the number of possible choices (codebooks) is higher than the $\eta = 1$ case.

\footnote{Note that the $E_8$ lattice is the best lattice among the known lattices for the dimension $m = 8$ [14].}
3.3 Adaptive Quantizer Scheme for H.263+

In this section, we summarize the quantizers in the ITU-T H.263+ from the documents [77] and [23]. We then propose the AQS, which adaptively employs a new quantizer, SAPQ, only for the DCT blocks that require large number of bits.

A. Quantizer in H.263+

In order to introduce the quantizer in the recommendation of H.263+ and the test model, we define the following:

\[
\begin{align*}
Q P &:= \text{quantization parameter}, \\
C O F &:= \text{a transform coefficient to be quantized}, \\
L E V E L &:= \text{the quantized version of the transform coefficient}, \\
R E C &:= \text{reconstructed coefficient value},
\end{align*}
\]

where the quantization parameter QP may take integer from 1 to 31. The quantizer step sizes are \(2 \cdot Q P\). QP = 4 for INTRA DC coefficients when not in the “advanced INTRA coding” mode [38, Annex I].

The inverse quantization reconstruction rule for all quantized coefficients can be expressed as

\[
R E C = \begin{cases} 
\text{sign}(L E V E L) \cdot R E C', & L E V E L \neq 0 \\
0, & \text{otherwise},
\end{cases}
\]

where

\[
R E C' = Q P \cdot (2 \cdot |L E V E L| + q) - p,
\]

and \(p = 0\) if QP is odd and \(p = q\) otherwise. \(q = 1\) for INTER coefficients and INTRA non-DC coefficients when not in the advanced INTRA coding mode. \(q = 0\) for INTRA DC coefficients when not in the advanced INTRA coding mode and for INTRA coefficient (DC and non-DC) when in the advanced INTRA coding mode. In this section, we will focus on the quantization of the INTER blocks. In other words, for the INTER coefficient case, \(q = 1\) in (3.9). On the other hand, the quantizer for INTER coefficients in the UBC (University of British Columbia) TMN8 (test model, near-term, version 8) [22] is given by

\[
L E V E L = \text{sign}(C O F) \cdot L E V E L',
\]

where

\[
L E V E L' = \lfloor (|C O F| - |Q P/2|)/(2 \cdot Q P) \rfloor,
\]

and \(\lfloor x \rfloor, x \in \mathbb{R},\) is the largest integer less than or equal to \(x\). Hence the decision levels are given by

\[
\cdots - 4Q P - |Q P/2|, -2Q P - |Q P/2|, 2Q P + |Q P/2|, 4Q P + |Q P/2|, \cdots.
\]

However, the reconstruction levels are

\[
-3Q P + p, 0, 3Q P - p, \cdots.
\]

Note that each non-zero reconstruction level is not the center of the corresponding quantizer region.

B. Adaptive Quantizer Scheme based on SAPQ

The optional "alternative INTER VLC" mode of H.263+ [38, Annex S] can improve the coding performance of the quantizer output when significant changes are evident in the picture. In the "busy" INTER blocks, which have a large number of large-magnitude coefficient, the option employs the INTRA
VLC table \[38, \text{TABLE I.2}\] of the advanced INTRA coding mode \[38, \text{Annex I}\]. Hence, we can reduce the required bits for the block and maintain the same picture quality. In this section, we propose an AQS for such busy blocks in order to reduce the quantizer distortion. In the busy blocks, we use non-zero level quantizers (quantizers that have no zero-value reconstruction level) to obtain gains based on the SAPQ theory. The proposed AQS can be applied to the current option, alternative INTER VLC mode.

In the proposed AQS, each block is first classified into two classes: non-busy block class and busy block class, based on the DCT coefficients of each block. An SAPQ is then applied to the busy blocks only to reduce the distortions of the busy blocks. Non-busy block uses the original quantizer. In the busy blocks, the SAPQ uses several non-zero level quantizers, where only the reconstruction levels are different from the original quantizers of H.263+.

We can consider the following two different methods of classification for finding the busy blocks:

1. Counting the required bits for a block and classifying by using the number of bits by setting a threshold \(TH\) for this number.

2. In the alternative INTER VLC mode \[38, \text{Annex S}\], the blocks that use the INTER VLC table are classified as the busy block.

Note that both these methods do not need any side information to indicate whether the current block is the busy block or not. However, by using the "quantizer information \(\text{(DQUANT)}\)\", we could arbitrarily indicate the busy blocks (and then use side information). The first method is a simple classifier without changing the VLC bit stream for the coefficients. If we consider the second method, then the proposed scheme can be applied to the option alternative INTER VLC mode as a modified form of the option. The current option is focussed on reducing the number of required bits. However, if this option is modified by using the AQS, then we can improve the SNR of the busy blocks as well as reduce the required bits.

We now introduce the quantizer part of the proposed AQS. In a DCT block, we additionally define the following:

\[
\begin{align*}
\text{COFI}_i & := \text{a transform coefficient to be quantized}, \\
\text{LEVEL}_i & := \text{the quantized version of the transform coefficient}, \quad \text{and} \\
\text{REC}_i & := \text{reconstructed coefficient value},
\end{align*}
\]

where \(i \in \{1, \ldots, 64\}\) is the index for the 64 DCT coefficients. We use the following quantization rule, which yields \(\text{LEVEL}_i\) for the VLC of the INTER coefficients;

\[
\text{LEVEL}_i = \text{sign(\text{COFI}_i)} \cdot \text{LEVEL}'_i, \tag{3.14}
\]

where \(\text{LEVEL}'_i = \lfloor (\text{COFI}_i - |QP/2|)/(2 \cdot \text{QP}) \rfloor\). In order to introduce the inverse quantization rule, following inverse quantizers are defined. The first inverse quantization rule, which is designated by \(\text{IQ}_0\), is used for the non-busy blocks.

\[
\text{IQ}_0: \quad \text{REC}_i = \begin{cases} 0, & \text{LEVEL}_i = 0 \\ \text{sign(LEVEL}_i) \cdot \text{REC}'_i, & \text{otherwise}, \end{cases}
\tag{3.15}
\]

for \(i = 1, \ldots, 64\), where

\[
\text{REC}'_i = \text{QP} \cdot (2 \cdot |\text{LEVEL}_i| + 1) - p,
\tag{3.16}
\]

and \(p = 0\) if \(\text{QP}\) is odd and \(p = 1\) otherwise. (Note that \(\text{IQ}_0\) is the same as the original inverse quantization rule for the INTER coefficients in (3.9). The recommendation of H.263+ specifies only this inverse quantization rule \[38, \text{p.411}\].) The next set of rules that describe the SAPQs that are used for the busy blocks in the proposed AQS. For \(j = 1, 2, \ldots, 2q\),

\[
\text{IQ}_j: \quad \text{REC}_{i,j} = \begin{cases} c_{i,j} \cdot \text{LEVEL}_i, & \text{LEVEL}_i = 0 \\ \text{sign(LEVEL}_i) \cdot \text{REC}'_i + d_{i,j}, & \text{otherwise}, \end{cases}\tag{3.17}
\]
for \(i = 1, \ldots, 64\), where
\[
REC'_i = QP \cdot (2 \cdot |LEVEL_i| + 1) + b_{i,j} - p,
\]
and \(p = 0\) if \(QP\) is odd and \(p = 1\) otherwise. In the inverse quantizer \(IQ_j\), \(b_{i,j}\), \(c_{i,j}\), and \(d_{i,j}\) are integers, and \(j (\eta\ bits)\) is the side information. (Note that these parameters are dependent on \(i\), the index for the DCT coefficient in a DCT block, and \(|b_{i,j}|\) and \(|c_{i,j}|\) should be even and \(|d_{i,j}|\) should be odd to ensure the odd values of \(REC_i\).) The design of the parameters and the discussion is introduced in Section 3.4.

We now introduce the SAPQ quantization rule. Let \(D_j\) denote the distance for a busy block that is inversely quantized by \(IQ_j\), where \(D_j := 64^{-1} \sum_{i=1}^{64} (COFi - REC_{i,j})^2\), for \(j = 1, 2, \ldots, 2^\eta\). The SAPQ is performed by finding the quantizer that yields the minimum distance among all \(D_j\). From (2.6), the smallest distance can be rewritten as
\[
\min_{j \in \{1, 2, \ldots, 2^\eta\}} \frac{1}{64} \sum_{i=1}^{64} (COFi - REC_{i,j})^2.
\]

The average of the distances of (3.19) represents the performance of the proposed AQS. The side information, which has a fixed length \(\eta\) bits, is added to the bit stream of the busy block after the last coefficient so that the bits are decodable.

### 3.4 Quantizer Design and Simulation

Designing an SAPQ quantizer in the proposed AQS of (3.17) is finding the appropriate parameters \(b_{i,j}\), \(c_{i,j}\), and \(d_{i,j}\). In this section, we introduce several designed quantizers and their numerical performance. The simulation was conducted based on the UBC TMN8 [23].

#### A. Classification

In this section, we use the first classification method that is addressed in Section 3.3. First, quantize \(COFi\) according to (3.14), and using the VLC tables for the INTER blocks [38, Tables 16,171 and \(LEVEL_i\), encode the quantizer output to a bit stream. Next, for each block, count the total bits required for the DCT coefficients (TCOFF) [38]. Note that, if \(LEVEL_i = 0\) for all \(i\) in a block, then using the "coded macroblock indication" (COD), the "macroblock type and coded block pattern for chrominance" (MBPC), and the "coded block pattern for luminance" (CBPY) bits, no further information is transmitted for the block. If for some \(i\), \(LEVEL_i \neq 0\) in a block, then this block is called coded block. In Fig. 3.2, the number of the coded blocks in each frame of a QCIF (quarter common intermediate format) video sequence, FOREMAN, is illustrated for 100 frames that were coded as the P-pictures. In this figure, \(TH = 10\) and \(TH = 20\) imply that the numbers of the coded blocks of the luminance blocks, which is called busy Y-blocks, have more or equal to 10 bits and 20 bits for TCOFF, respectively. In the Fig. 3.2 case, we note that 11% of the total DCT blocks are coded blocks. (Note that a QCIF frame has 594 DCT blocks.) For an appropriate threshold \(TH\), we classify the luminance blocks into the busy and non-busy Y-blocks. (For example, the blocks have more than or equal to \(TH = 20\) are the busy Y-blocks.) For several values of \(TH\), the peak signal to noise ratios (PSNRs) of the busy Y-blocks are summarized in Tables 3.4 and 3.2, where the signal variance is given by \(255^2\). Note that the main approach of the proposed AQS is using the SAPQ only for the busy Y-blocks to decrease the quantizer distortions of the blocks. Since the chrominance blocks usually require small amount of bits, the expected gain from the SAPQ will be negligible. (Note that the uniform SAPQ obtains a gain from the space-filling advantage.) However, for the chrominance blocks, instead of using SAPQ, simply decreasing \(QP\) can obtain good performance in a similar manner to the "modified quantization mode" [38, Annex T].
Figure 3.2: Number of coded blocks in each frame of a QCIF video sequence, FOREMAN at QP = 16. (TH = 10: Y-blocks that requires more or equal to 10 bits, and TH = 20: Y-blocks that requires more or equal to 10 bits.)

Table 3.1: PSNR's (dB) of the Busy Y-blocks and the Average Number of Busy Blocks Per Picture

<table>
<thead>
<tr>
<th>FOREMAN QCIF Video Sequence (100 Pictures by TMN8 at QP = 16)</th>
<th>PSNR: Y = 29.81 dB, CB = 35.30, CR = 35.74, Average Number of Bits = 1920 Bits/Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR (dB)</td>
<td>Busy Y-blocks/Frame</td>
</tr>
<tr>
<td>TH</td>
<td>10</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>26.41</td>
<td>32.9</td>
</tr>
</tbody>
</table>

B. SAPQ Design and Results

As discussed in Section 3.3, the quantizer for the INTER block is a modified uniform quantizer having a dead-zone. Hence, we use different parameters $c_{i,j}$ and $d_{i,j}$ for the zero level and the non-zero level shifting, respectively. (Note that the usual lattice VQ cannot implement this kind quantizer.) Using the CR notation $r_j^i$ defined after (3.1), we let

$$ c_{i,j} = \begin{cases} 
DEV_1 \cdot (4 \cdot r_j^i - 1), & \text{for } i = 1, \ldots, m \\
0, & \text{for } i = m + 1, \ldots, 64,
\end{cases} $$

(3.20)

for $j = 1, 2, \ldots, 2^n$, where a non-negative integer $DEV_1$ is a shifting constant for the zero level in the SAPQ codebooks, and $m \leq 64$. Note that $c_{i,j} \in \{-DEV_1, DEV_1\}$ for $i = 1, \ldots, m$ and $j = 1, 2, \ldots, 2^n$.

In the same manner, let

$$ d_{i,j} = \begin{cases} 
DEV_2 \cdot (4 \cdot r_j^i - 1), & \text{for } i = 1, \ldots, m \\
0, & \text{for } i = m + 1, \ldots, 64,
\end{cases} $$

(3.21)

for $j = 1, 2, \ldots, 2^n$, where $DEV_2$ is a non-negative integer. The parameter $b_{i,j}$ is related to the reconstruction value, which is defined as

$$ b_{i,j} = \begin{cases} 
DEV_3, & \text{for } i = 1, \ldots, m \\
0, & \text{for } i = m + 1, \ldots, 64,
\end{cases} $$

(3.22)
Table 3.2: PSNR’s (dB) of the busy Y-blocks and the average number of busy blocks per P picture

SUZIE QCIF video sequence (100 P pictures by TMN8)

<table>
<thead>
<tr>
<th>PSNR (dB)</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Busy Y-blocks</td>
<td>16.4</td>
<td>5.1</td>
<td>1.8</td>
<td>0.6</td>
</tr>
</tbody>
</table>

PSNR: Y = 32.27 dB, Cb = 39.49, Cr = 39.00. Average number of bits = 1186 bits/frame

111 coded blocks/frame: 27.06 dB

Figure 3.3: Quantizer reconstruction levels of IQ in the SAPQ. (IQ, for i = 1, ..., m in the SAPQ based on CR-D_{m_i}).

where DEV_3 is an integer.

First, we observe the performance of the proposed AQS that is designed based on the lattice D_{n_i}. If we consider the lattice D_{8_i}, then \eta = 1, and \eta = 2, and \delta = 1, and \delta = 2 are obtained from the two CRs, which are given in (3.3). The quantizer levels of this SAPQ are illustrated in Fig. 3.3, where we suppose that DEV_2 < DEV_3, and QP is odd. In Fig. 3.3, the first codebook (j = 1) has the negative value -DEV_1 instead of the zero level and the second codebook (j = 2) has the positive value DEV_1 instead of the zero level. Consider only 8 coefficients as shown in Fig. 3.4, from the 8 x 8 DCT block. Suppose that the numbers in Fig. 3.4 are indexed by i. For the other 56 coefficients, we use the inverse quantizer in (3.14) even when the block is classified into a busy block. In Tables 3.4 and 3.4, for DEV_3 = 4 and DEV_3 = 2, respectively, we provide numerical results of the proposed AQS with respect to DEV_1 and DEV_2.

Table 3.3: Proposed AQS with respect to the Parameters

FOREMAN QCIF Video Sequence (100 P Pictures by TMN8 at QP = 16)

CR-D_{8_i} (m = 8), \eta = 1, TH = 20, DEV_3 = 4.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>PSNR (dB)</th>
<th>Busy Y-block PSNR (dB)</th>
<th>Average bits/frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEV_1</td>
<td>DEV_2</td>
<td>Y</td>
<td>Cb</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>29.84</td>
<td>35.19</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>29.79</td>
<td>35.34</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>29.83</td>
<td>35.20</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>29.84</td>
<td>35.27</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>29.81</td>
<td>35.24</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>29.82</td>
<td>35.27</td>
</tr>
<tr>
<td>TMN8</td>
<td>29.81</td>
<td>35.30</td>
<td>35.74</td>
</tr>
</tbody>
</table>
these simulations, the side bit, $q$, is added to the next of the last TCOFF so that the decoder can decode
the side bit, where the total bits of each frame is aligned with bytes (see TMN8), and the first frame is
the I picture with QP and the next 100 frames are P-pictures at the same QP. For various values of TH,
the results are summarized in Table 3.4. We note that, for $TH = 10 \sim 40$, the gains are approximately
constant around 0.1 dB. These gains increase if we increase the side bits $q$ by employing CR-E8. For
each $D_R^m$, CR-1, CR-3, and CR-E8 of section 2.2, the simulation results are summarized in Table 3.4.
From this table, we can also notice the gains for several different video sequences. From Tables 3.4 \textendash} 3.4,
we note that the overall performance of the AQS and TMN8 are very similar in terms of PSNR and bit
rate. However, the AQS shows a better PSNR in the busy Y-blocks. Furthermore, from these tables, we
can find that the appropriate values of $DEV_1$, $DEV_2$, and $DEV_3$ for the QP = 16 case are $DEV_1 = 7$,
$DEV_2 = 2$, and $DEV_3 = 4$.

For different values of the quantization parameter QP, we define the parameters, $DEV_1$, $DEV_2$, and
$DEV_3$ as follows.

$$DEV_1 = [QP/2] - p,$$

(3.23)

where $p = 0$ if $[QP/2]$ is odd and $p = 1$ otherwise,

$$DEV_2 = [QP/8] - p,$$

(3.24)

where $p = 0$ if $[QP/8]$ is even and $p = 1$ otherwise, and

$$DEV_3 = [QP/4] - p,$$

(3.25)
Table 3.5: Proposed AQS with respect to parameters
SUZIE QCIF video sequence (100 P pictures by TMN8 at QUANT = 16)
\( m = 8 \) (8 coefficients in Fig. 3.4(a)), \( \eta = 1, TH = 20, b_{i,j} = 4 \) for all \( i \) and \( j \).

<table>
<thead>
<tr>
<th>Parameters</th>
<th>( DEV_i )</th>
<th>( DEE )</th>
<th>( Y ) PSNR (dB)</th>
<th>( C_b ) PSNR (dB)</th>
<th>( C_r ) PSNR (dB)</th>
<th>Busy Y block PSNR (dB)</th>
<th>Average bits/frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td>32.28</td>
<td>39.39</td>
<td>39.13</td>
<td>26.93</td>
<td>1193</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>32.28</td>
<td>39.28</td>
<td>39.26</td>
<td>27.03</td>
<td>1193</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>32.26</td>
<td>39.45</td>
<td>39.01</td>
<td>27.13</td>
<td>1194</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>32.28</td>
<td>39.48</td>
<td>39.25</td>
<td>27.07</td>
<td>1188</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>32.27</td>
<td>39.45</td>
<td>39.09</td>
<td>26.96</td>
<td>1192</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>32.28</td>
<td>39.51</td>
<td>39.04</td>
<td>27.04</td>
<td>1192</td>
<td></td>
</tr>
<tr>
<td>TMN8</td>
<td></td>
<td>32.27</td>
<td>39.49</td>
<td>39.00</td>
<td>26.79</td>
<td>1186</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.6: Proposed AQS with respect to TH
FOREMAN QCIF video sequence (100 P pictures by TMN8 at QUANT = 16)
\( CR-DK \) (\( m = 8 \)), \( \eta = 1, DEV_i = 7, DEE = 2 \), \( DEV_3 = 4 \).

<table>
<thead>
<tr>
<th>TH</th>
<th>PSNR (dB)</th>
<th>Busy Y-block PSNR (dB)</th>
<th>Average bits/frame</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( Y )</td>
<td>( C_b )</td>
<td>( C_r )</td>
</tr>
<tr>
<td>10</td>
<td>29.83</td>
<td>35.07</td>
<td>35.82</td>
</tr>
<tr>
<td>20</td>
<td>29.84</td>
<td>35.27</td>
<td>35.71</td>
</tr>
<tr>
<td>30</td>
<td>29.81</td>
<td>35.30</td>
<td>35.65</td>
</tr>
<tr>
<td>40</td>
<td>29.83</td>
<td>35.29</td>
<td>35.70</td>
</tr>
<tr>
<td>TMN8</td>
<td>29.81</td>
<td>35.30</td>
<td>35.74</td>
</tr>
</tbody>
</table>

where \( p = 0 \) if \( \lfloor QP/4 \rfloor \) is even and \( p = 1 \) otherwise. The results with respect to QP is shown in Tables 3.4. We observe that the average bits per frame increase slightly as QP gets small.

In Fig. 3.5, an example of required bit change in the proposed AQS is illustrated for a small quantization parameter, QP = 8. For the comparison, we introduce a modified H.263, where QP − 1 is used for the busy blocks (Fig. 3.5(a)). Hence, we can also obtain a good performance for the busy blocks. However, the total bits are increased by 9.1%, whereas 3.5% in the SAPQ case of \( \eta = 4 \) and \( m = 8 \). Further, the variance of the bit change is higher than the SAPQ case as shown in Fig. 3.5. For the SAPQ case, the bit changes are nearly close to the original case.

C. Subjective Comparison

In this section, we compare the performance of the proposed AQS by displaying the reconstructed video sequences. When we compare the reconstructed video sequences, we could consider the following comparison criteria:

1. blocking artifacts between adjacent blocks,
2. smeared/noisy block, and
3. inconstant blocks between frames.

In order to improve the video quality based on the criteria, we should improve the all techniques of the coder, such as motion vector estimation (the loop filter) and the quantizer. In other words, since the blocks in H.263+ are classified into coded and non-coded blocks, i.e., the blocks to be quantized and not quantized, simply increasing the quantizer performance for the busy blocks might result more blocking.
artifacts especially for large values of the quantization parameter QP. For example, while we compare the two reconstructed video sequences, which are encoded by H.263+ for QP = 15 and QP = 16, respectively, we could not determine which one was better than the other one if we only consider the blocking artifacts. Increasing the quantizer performance is more closely related with the second criterion, "smeared/noisy block". Decreasing QP will obviously reduce the smear and noise inside the block. However, sometimes, decreasing QP enforces several blocks to be quantized, which were non-coded blocks before decreasing QP. Hence the blocks could be noisy blocks due to a marginal quantization between coded and non-coded blocks especially for large QPs. Since the performance of the current frame is dependent on the reference frame, we can observe a similar effect even when QP is fixed if we were to change the quantizer scheme. Even though the block is noisy, however, we can alleviate the smear effect of block. The noisy blocks could be the main reason for the inconstancy between frames. Therefore, since we only increase the quantizer performance for the busy Y-blocks in the proposed AQS, we focus on the second criterion to compare the reconstructed video sequences.

For the reference video sequences for comparison, we consider two reconstructed video sequences for a given value of QP; TMN8s with QP and QP - 1. The same parts of the images from the reconstructed video sequences are illustrated in Figs. 3.6 and 3.7 for frame 10 and frame 19, respectively, to show the improvement based on the smear/noise blocks criterion. In these figures,

<table>
<thead>
<tr>
<th>Table 3.7: Proposed AQS with Respect to Video Sequences and η</th>
</tr>
</thead>
<tbody>
<tr>
<td>QCIF</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>FOREMAN</td>
</tr>
<tr>
<td>QP = 15</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TMN8</td>
</tr>
<tr>
<td>SUZIE</td>
</tr>
<tr>
<td>η = 1 (CR-Δ2)</td>
</tr>
<tr>
<td>η = 2 (CR-1)</td>
</tr>
<tr>
<td>η = 3 (CR-3)</td>
</tr>
<tr>
<td>η = 4 (CR-Δ8)</td>
</tr>
<tr>
<td>TMN8</td>
</tr>
<tr>
<td>CARPHONE</td>
</tr>
<tr>
<td>η = 1 (CR-Δ2)</td>
</tr>
<tr>
<td>η = 2 (CR-1)</td>
</tr>
<tr>
<td>η = 3 (CR-3)</td>
</tr>
<tr>
<td>η = 4 (CR-Δ8)</td>
</tr>
<tr>
<td>TMN8</td>
</tr>
<tr>
<td>CLAIRE</td>
</tr>
<tr>
<td>η = 1 (CR-Δ2)</td>
</tr>
<tr>
<td>η = 2 (CR-1)</td>
</tr>
<tr>
<td>η = 3 (CR-3)</td>
</tr>
<tr>
<td>η = 4 (CR-Δ8)</td>
</tr>
<tr>
<td>TMN8</td>
</tr>
</tbody>
</table>
TABLE 3.8: PROPOSED ADAPTIVE CODING SCHEME WITH RESPECT TO QP
FOREMAN QCIF VIDEO SEQUENCE (100 P PICTURES BY TMN8)
CR-1 \((m = 8), \eta = 2, TH = 20\)

<table>
<thead>
<tr>
<th>QP</th>
<th>PSNR (dB)</th>
<th>Busy Y-block PSNR (dB)</th>
<th>Average bits/frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>AQS TMN8</td>
<td>37.00 40.23 41.81 36.61</td>
<td>13511</td>
</tr>
<tr>
<td>8</td>
<td>AQS TMN8</td>
<td>33.10 37.53 38.67 30.99</td>
<td>5012</td>
</tr>
<tr>
<td>12</td>
<td>AQS TMN8</td>
<td>31.11 36.09 36.94 27.98</td>
<td>2801</td>
</tr>
<tr>
<td>16</td>
<td>AQS TMN8</td>
<td>29.85 35.27 35.75 25.94</td>
<td>1919</td>
</tr>
<tr>
<td>20</td>
<td>AQS TMN8</td>
<td>28.88 34.68 34.80 24.40</td>
<td>1513</td>
</tr>
<tr>
<td>24</td>
<td>AQS TMN8</td>
<td>28.15 34.03 34.21 23.10</td>
<td>1291</td>
</tr>
<tr>
<td>28</td>
<td>AQS TMN8</td>
<td>27.62 33.50 33.40 22.09</td>
<td>1154</td>
</tr>
</tbody>
</table>

where the PSNRs are obtained for reconstructed 100 P-pictures of FOREMAN QCIF video sequences.

In Figs. 3.6 and 3.7, we observe that (b) has the best quality in the criterion of the smear/noise criterion, i.e., (b) shows the clearest shape among the figures even though there are more blockiiig artifacts. Focus on his right collar in the figures. In both Figs. 3.6 and 3.7, (a) and (c) have a similar blocking artifact and noise in the area. However, (d) shows a similar shape of that of (b). Through the reconstructed video sequences, we can find that the quality of the (d) case is closer to the (b) case than the (a) case. For the \(\eta = 2\) case of (c) has a similar quality of that of (a). However, a variation of the case show a better visual quality as shown in Fig. 3.9(a). The coset representatives CR-1 is modified to

\[
\begin{align*}
\text{CR-1}': & \quad r_1' = (0, 0, 0, 0, 0, 0), \quad r_2' = 1/2(1, 1, 1, 1, 1), \\
& \quad r_3' = 1/2(1, 0, 1, 1, 0, 0), \quad r_4' = 1/2(0, 1, 0, 0, 0, 0).
\end{align*}
\]

The TCOFFs for the variation is the 6 coefficients in Fig. 3.8, where \(m = 6\). In this numerical result for 100 P-pictures, note that the PSNR of the luminance component is 29.84 dB at 1921 bits/frame and the PSNR of the busy Y-blocks is 25.95 dB, which is slightly less than the variation.

We can consider a simple AQS by modifying the TMN8 of QP, where the quantization parameter QP - 1 is used for the busy Y-blocks. We can obtain a similar performance of the proposed AQS. However, for small quantization parameters, we should increase the bits much higher than the proposed AQS case. We could also combine the SAPQ and the modified TMN8 scheme.

D. Non-Zero Level Quantizers

If we quantize a positive valued signal, then it seems that there is no difference in using the mid-tread or mid-rise type quantizer, where the mid-tread type has the zero level in the output levels, and the mid-rise type does not have the zero level. However, if we consider a quantizer for the DCT coefficients or the residual signals in the differential pulse code modulation (DPCM), then the inputs to be quantized have a high population near zero. (Note that the distributions of these inputs are usually regarded as the Laplacian probability density function [68].) Hence, in this case, it is important to select a quantizer that has the zero level. However, if the distribution of the input is symmetrical and the quantizer output
is encoded by fixed binary codes, then the number of levels of the quantizer is a power of 2, i.e., even. Hence, in order to obtain a minimum quantizer distortion, the mid-rise type is usually employed, since the mid-rise type quantizer have an even number of symmetrical levels [32],[70]. However, we might observe the granular noise at smooth areas especially in the DPCM systems due to the quantizer that does not have the zero level. If we use the entropy coder, such as VLC, for the quantizer outputs, then we can use a mid-tread type quantizer. However, in order to obtain more gain from the run-length coding, the quantizer for the INTER block has the wide quantizer region, called dead zone. Even though we can reduce the required bits by extending the dead zone, the image quality will be degraded, since a large portion of the input is concentrated around zero, but the quantizer step size is relatively large. Unfortunately, many meaningful non-zero inputs are at the zero level. Using the proposed SAPQ, the non-zero level quantizers, which have small values near zero but have a dead zone in the SAPQ could alleviate this problem.

### 3.5 Conclusion

In this section, for the ITU-T H.263+, we proposed an adaptive quantizer scheme (AQS) by employing the new quantizer scheme, sample-adaptive product quantizer (SAPQ). The proposed AQS shows a good performance for the range of the quantization parameter QP and different types of the video sequences. In the busy blocks, the SNR is increased by 0.1 ~ 0.5 dB. However, in the overall performance, the proposed AQS has produced the total SNR and bit rate that are equal to or slightly better than in the original scheme. It should be noted that SAPQ can achieve a VQ-level performance at high bit rates with a low encoding complexity of scalar quantizers. Since SAPQ has a scalar quantizer structure, we can easily apply SAPQ to the current coding systems in order to obtain a VQ performance as shown in the proposed AQS. Hence, the SAPQ can be combined with any of the current options in H.263+. For example, it is possible to adopt SAPQ in the optional alternative INTER VLC mode (Annex S).

---

3 The quantization in the DCT-based image compression system, which does not employ the entropy coder, is usually based on the mid-rise nonuniform quantizers with a bit allocation technique [36].
Figure 3.6: Frame 10 from the reconstructed **FOREMAN** QCIF video sequence. (a) TMN8 at $QP = 16$. (b) TMN8 at $QP = 15$. (c) AQS at $QP = 16$ and $\eta = 2$. (d) AQS at $QP = 16$ and $\eta = 4$. 
Figure 3.7: Frame 19 from the reconstructed FOREMAN QCIF video sequence. (a) TMN8 at $QP = 16$. (b) TMN8 at $QP = 15$. (c) AQS at $QP = 16$ and $\eta = 2$. (d) AQS at $QP = 16$ and $\eta = 4$. 
Figure 3.8: $8 \times 8$ DCT block and the 6 coefficients for SAPQ ($m = 6$).

Figure 3.9: AQS at $QP = 16$, $\eta = 2$, and $m = 6$. (a) Frame 10. (b) Frame 19.
4. Generalized SAPQ for k-Dimensional Vectors

4.1 Generalization of SAPQ

In this section we extend the SAPQ in Section 2 to the k-dimensional vectors $X_1, \ldots, X_k$, as the discrete-time source to be quantized. Here $X_i := (X_{i1}, \ldots, X_{ik})$ is a random vector in $\mathbb{R}^k$ and $m$ is the sample size or the adaptation period. Suppose that $E\|X_i\|_r < \infty$, for $i = 1, \ldots, m$, where $\| \cdot \|_r$ denotes the $r$th power of the $l_n$ norm to be used for the distortion measure. Let $C_n$ denote the class of sets that take $n$ points from $\mathbb{R}^k$, and let the sets in $C_n$ be called "n-level codebooks", where each such codebook has $n$ codewords. Denote the block length $L := km$ as the quantizer dimension, and let an observation of $X_1, \ldots, X_k$ be denoted by $X^w_1, \ldots, X^w_m$, where $w$ is a sample point of the underlying sample space $\Omega$; we call this observation a sample.

For every sample, SAPQ employs codebook sequences from a previously designed set of $N$ codebook sequences available at both the encoder and the decoder. In SAPQ, it is important; to note that the codebook sequences can be changed adaptively for each sample that contains $m$ random vectors. Let $C_{i,j} \subset \mathbb{R}^k$ denote the $j$th codebook for each $X_i$, where $j \in \{1, \ldots, N\}$, $N := 2^\eta$, and $\eta$ is a non-negative integer. Assume that the samples of $X_1, \ldots, X_k$ sequentially enter the encoder. The adaptive scheme quantizes each sample $X^w_1, \ldots, X^w_m$ using the codebooks $C_{1,j}, \ldots, C_{m,j}$ to form the $N$ candidates of the $m$-codebook sample distances defined by

$$\frac{1}{L} \sum_{i=1}^{m} \min_{y \in C_{i,j}} \|X^w_i - y\|^r, \text{ for } j = 1, \ldots, N. \quad (4.1)$$

Here, we suppose that $C_{i,j} \in C_n$, for $j = 1, \ldots, N$, where $n \in \mathbb{N}$. The distance in (4.1) is a random variable defined on the underlying sample space if $X^w_1, \ldots, X^w_m$ is replaced with the random vector $X_1, \ldots, X_k$. Note that, for a fixed $j$, in order to quantize the $m$ random vectors, a sequence of $m$ codebooks $C_{1,j}, \ldots, C_{m,j}$ are employed as shown in the $m$-codebook sample distance. For each sample, the adaptive scheme finds a codebook sequence, from a finite class of codebook sequences, that yields the minimum distance given by (4.1). The resultant distortion of SAPQ, given by taking expectations in (4.1), is

$$: = E \left\{ \min_j \frac{1}{L} \sum_{i=1}^{m} \min_{y \in C_{i,j}} \|X_i - y\|^r \right\} \quad (4.2)$$

We call the SAPQ in (4.2), which is based on the $m$-codebook sample distance, $m$-SAPQ.

In SAPQ, for each sample, the encoder transmits bits, for the codebook index with $m$ quantized element indices, in the form of a feed-forward adaptive scheme. This makes it possible to replace different codebook sequences for each sample of $X_1, \ldots, X_k$. In other words, the encoder quantizes $m$ vectors of a sample $X^w_1, \ldots, X^w_m$ using a codebook sequence of size $m$ from $2^\eta$ codebook sequences and replaces the codebook sequence for each sample. Therefore the total bit-rate in $m$-SAPQ is given by

$$R = \frac{1}{L} \log_2 \prod_{i=1}^{m} n_i' + \frac{\eta}{L}, \quad (4.3)$$

where $\eta$ are the additional bits required in our scheme as side bits to indicate which codebook sequence is employed.

Note that $m$-SAPQ requires at most $mN$ different codebooks from $C_n$. Hence, if $m$ is large, the decoder needs a large memory for the codebooks and the codebook design complexity may be high. In order to reduce the required number of codebooks, one possibility is to use the 1-codebook sample distance defined by

$$\frac{1}{L} \sum_{i=1}^{m} \min_{y \in C_{i,j}} \|X^w_i - y\|^r, \text{ for } j = 1, \ldots, N \quad (4.4)$$
Note that for each distance, we use only one codebook $C_j$. Here we assume that $C_j \in \mathcal{L}_{u'}$, for all $j$, and $u' \in \mathbb{N}$.

The resultant distortion of this simplified SAPQ is given by

$$D_{\text{SAPQ}} := E \left\{ \min_j \frac{1}{L} \sum_{i=1}^{m} \min_{y \in C_j} \|x_i - y\|^r \right\}$$  \hspace{1cm} (4.5)

We call this SAPQ, which is based on the 1-codebook sample distance, 1-SAPQ. Note that the bit-rate for 1-SAPQ is given by $R = (\log, n')/k + \eta/L$.

If we consider the quantizer dimension $L$, then the SAPQs are structurally constrained VQs in $L$-dimensions, where the SAPQs are using $m_k$-dimensional VQs. Note that directly implementing the $L$-dimensional VQ performance with the order of $k$-dimensional VQ computational complexity, where $k < \kappa < L$. Due to the VQ encoding complexity, the block size is limited to a size of $4 \times 4$ commonly in images. Hence, employing SAPQ is very useful in quantizing the $32 \times 32$, $16 \times 16$, or $8 \times 8$ image blocks in the segmentation based codecs. Note that, for such blocks, currently, DCT and the product VQ scheme is used. In the Nokia's MVC video codec proposal, note that DCT and a product VQ is used for the $8 \times 8$ image block.

### 4.2 SAPQ for Gauss-Markov Sources

If we use the simple codebook generation algorithm and the split method for the initial codebook, then the performance for correlated sources is not so good. In other words, we cannot obtain the expected gains. SAPQ codebook design for the correlated sources are very dependent on the selected initial codebook as shown in Fig. 2.3. Further, the selection is getting difficult as $n'$ gets large. For a Gaussian i.i.d. source, the distortion of 1-SAPQ is $-6.93$ dB. However, for a Gaussian-Markov source (1st order Markov with the correlation coefficient 0.9), it is reduced to $-8.07$ dB for the split constant $\kappa = 0.01$ and $-8.73$ dB for the split constant $\kappa = 1$, respectively, as shown in Fig. 2.3. In this case, we can use the generalized SAPQ for some values of $k > 1$. In Fig. 4.1, the $k = 1$ and $k = 2$ cases of 1-SAPQ are compared for a Gaussian i.i.d. and a Gauss-Markov source. If we use a SAPQ of $k = 2$, rather than $k = 1$, we can achieve more gains for both the source types. However, the improvement for the Gauss-Markov source case is higher than the Gaussian i.i.d. case. This fact is very important in quantizing image blocks. If there exits high correlation only inside a small part of a image block, then we choose an appropriate vector dimension according to the small part, which is usually smaller than the image block size. This kind image block is an active region, and the block size is relatively small like $4 \times 4$.

In Fig. 4.1, we use two different split constants $\epsilon = 0.001$ and 1. For the $\epsilon = 1$ case, we can usually obtain more gains or faster convergence rate in the SAPQ design algorithm. For example, focus on the 1-SAPQ at $\eta = 3$ ($R = 2.1875$) for the Gauss-Markov source case. The resultant distortions are the almost same as $-16.12$ dB. However, the number of iterations for $\epsilon = 0.001$ and $\epsilon = 1$ were 286 and 162, respectively. Here the distortion threshold in the SAPQ design algorithm is 0.0001. Regarding the SAPQ algorithm, we should conduct more research to achieve better local optimum and fast convergence for the real data.

### 4.3 Image Vector Quantization

In this section, we first the performance of SAPQ to standard VQ. In this case, the SAPQ framework uses a vector quantizer, i.e., $k > 1$. The channel noise effect on SAPQ is then observed for the real images.

We consider the vector quantization of a $512 \times 512$ image having $8 \text{b/pixel}$. In order to vector quantize images at a bit-rate of $1.0 \text{b/pixel}$, we consider a 4-dimensional VQ for $2 \times 2$ image subblocks; we need
Figure 4.1: Improvement of performance by increasing $k$ for a Gaussian i.i.d. and a Gauss-Markov source (1st order Markov with the correlation coefficient 0.9). (1-SAPQ results are obtained by varying $\eta$ from 1 to 4, where the split constant $\epsilon = 0.001$, otherwise specified.)
Figure 4.2: Ten 512 x 512 images with 8 b/pixel for TS (From the left upper image, Peppers, Airforcebase, Cablecar, Crowd, Diving, Einstein, Fruit, Lighthouse, Baboon, and Trains).

Figure 4.3: Image subblocks for 1-SAPQ. (a) $k = 2 \times 2$ and $m = 2 \times 2$. (b) $k = 2 \times 2$ and $m = 4 \times 1$. 

only 16 codewords in \( \mathbb{R}^4 \). Note that, if we were to increase the vector dimension (image subblock) to 4 x 4 in order to improve quantization performance, then the VQ needs \( 2^{16} = 65,536 \) codewords in \( \mathbb{R}^{16} \), which is a very large codebook for practical situations. We would also need a large memory of 1,048,576 scalar values for the codebook. Further, in order to generate the codebook using a TS, we would need more than 4,000 images for the TS if we set the training ratio, TS size to the codebook size, e.g., 1,000 (this would be virtually impossible to achieve in a practical situation). In this case, using the tree-structured VQ (TSVQ), TSVQ reduces the computational complexity in encoding, but cannot reduce the required number of codewords and the TS images. On the other hand, we can easily use an SAPQ (with \( k = 4 \)), having the quantizer dimension \( L = 16 \), since this would involve only a linear increase with respect to the power of side bits, of the number of codebooks and TS size (Table 4.2). Hence, for the same bit-rate as in the VQ case, employing SAPQ for \( k = 4 \) and \( L = 16 \), we can achieve gains over the 4-dimensional VQ, as shown in Table 4.1. In this table, two different grouping methods of the 2 x 2 image subblocks (as shown in Fig. 4.3(a) and (b)) for SAPQ are used. From the SAPQ results in Table 4.1, we can see that the two different grouping methods have distortion levels that are very close to each other, since we use the simple initial codebook finding method, the split method. However, we use the first grouping method \( m = 2 \times 2 \) in Fig. 4.3(a), since it seems to be better from a perceptual perspective. Note that the ten \( 512 \times 512 \) images, which are used for the TS, are shown in Fig. 4.2, and two images in Fig. 4.4 are used as the validating sequence (VS). In Fig. 4.5, several reconstructed images of Table 4.1 are illustrated to show the performance of the quantized images by SAPQ. The SAPQ shows more clearly reconstructed images in the edge areas. In the smooth areas, the contouring artifacts in the VQ case cannot be observed in the SAPQ case.

### 4.4 Performance Comparison and Trellis Coded Quantization

In Table 4.2, the traditional VQ and TSVQ are compared with several SAPQs in terms of their distortions (dB), number of multiplications for encoding, and required memory for the codewords. In this table, VQ and TSVQ are designed using GLA, where in the TSVQ, the breadth is equal to 2 [25]. For the quantizer dimension 2 (\( L = km = 2 \) in Table 4.2), the VQ distortions show the lowest ones for each bit-rates. However, the required multiplications and memory are very huge compared to those of 1-SAPQ. In the TSVQ case, TSVQ can reduce the multiplications, but, requires even more memory than the traditional VQ case. On the other hand, the 1-SAPQs can obtain similar performance, but the required multiplications and memory is very small. In the 1-SAPQ, we can obtain more gains by increasing \( m \) to 4 and 8. However, for small codebook size cases, e.g., \( n' = 2 \) at \( m = 4 \) and \( m = 8 \), the distortions are even worse than the 1-SAPQ case of \( L = 2 \) (the theoretical background on this fact and discussion are not shown in this report.) On the other hand, in this \( n' = 2 \) case, m-SAPQ can achieve gains over VQ, TSVQ, and 1-SAPQ (See Table 4.2). Similar simulations are performed on a Laplacian i.i.d. source, and summarized in Table 4.3. Further, in Table 4.4, the 1-SAPQ is compared with PQ, and L-dimensional VQ and TSVQ at a bit-rate of 0.5, where \( k \neq 1 \). In the Gaussian d.f. case, the 1-SAPQ distortions are very close to those of the L-dimensional VQ.

Several examples on the codebook size are illustrated in Fig. 4.6(a). From this Figure, we can see that the number of codewords for the VQ cases increase faster than the SAPQ cases as the bit-rate increases. In Fig. 4.6(b), the performance of SAPQ and VQ are compared at similar codebook size cases as shown in Fig. 4.6(a). The distortion of the SAPQ with \( m = 2 \) is bounded by the 2-dimensional VQ as discussed in Section II B. However, even though the SAPQ is a structurally constrained VQ of the 2-dimensional VQ, the SAPQ distortions is very close to those of the 2-dimensional VQ. Further, the codebook size is very small comparing to the 2-dimensional VQ as shown in Fig. 4.6(a). Furthermore, at similar levels of complexity with the 2-dimensional VQ, the SAPQ results shows better performance as shown in Fig. 4.6(b).

In Fig. 4.7, 1-SAPQ is compared with the different coding schemes, TCQ [54] and the entropy coded quantization [39],[18]. TCQ uses the Viterbi algorithm to encode memoryless sources, and uses a sliding
Figure 4.4: Original images (512 x 512, 8 b/pixel). (a) Lena. (b) Bridge.

Table 4.1: PSNR (dB) Comparison of Image Vector Quantization using SAPQ at 1.0 b/pxel and $\eta = 4$

<table>
<thead>
<tr>
<th>VS image</th>
<th>VQ $k' = 2 \times 2$</th>
<th>VQ $k = 2 \times 2, m = 2 \times 2$</th>
<th>SAPQ $k = 2 \times 2, m = 4 \times 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>29.8</td>
<td>31.8</td>
<td>31.2</td>
</tr>
<tr>
<td>Bridge</td>
<td>25.5</td>
<td>26.6</td>
<td>26.7</td>
</tr>
<tr>
<td>TS</td>
<td>25.7</td>
<td>27.2</td>
<td>27.1</td>
</tr>
</tbody>
</table>
Figure 4.5: Reconstructed images at 1.0 b/pixel. (a) Lena, VQ with $k' = 2 \times 2$ (29.8 dB). (b) Lena, SAPQ with $k = 2 \times 2$ and $m = 2 \times 2$ (31.8 dB). (c) Bridge, VQ with $k' = 2 \times 2$ (25.5 dB). (d) Bridge, SAPQ with $k = 2 \times 2$ and $m = 2 \times 2$ (26.6 dB).
### Table 4.2: Distortion (dB) and Complexity Comparison of SAPQ for Gaussian i.i.d. (Unit Variance) at k = 1

<table>
<thead>
<tr>
<th>Quan. Dim. (L)</th>
<th>Quantizer</th>
<th>Codebook Size (v)</th>
<th>Distortion (dB)</th>
<th>Multiplications</th>
<th>Memory (m)</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>VQ</td>
<td></td>
<td>-6.94</td>
<td>8</td>
<td>5</td>
<td>$2d$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-12.4</td>
<td>32</td>
<td>128</td>
<td>$2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-18.1</td>
<td>128</td>
<td>512</td>
<td>$2n^2$</td>
</tr>
<tr>
<td>2</td>
<td>TSVQ (Breadth = 2)</td>
<td>Depth (d)</td>
<td>-6.23</td>
<td>7</td>
<td>3</td>
<td>$2d$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-11.7</td>
<td>3</td>
<td>5</td>
<td>$2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-17.5</td>
<td>2</td>
<td>7</td>
<td>$2n^2$</td>
</tr>
<tr>
<td>2</td>
<td>1-SAPQ ($\eta = 1$)</td>
<td>Codebook Size (n')</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>$2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-6.91</td>
<td>8</td>
<td>4</td>
<td>$2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-12.3</td>
<td>4</td>
<td>8</td>
<td>$2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-17.9</td>
<td>4</td>
<td>8</td>
<td>$2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-23.7</td>
<td>2</td>
<td>16</td>
<td>$2n^2$</td>
</tr>
<tr>
<td>4</td>
<td>1-SAPQ ($\eta = 2$)</td>
<td>Codebook Size (n')</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>$2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-6.63</td>
<td>8</td>
<td>8</td>
<td>$2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-12.6</td>
<td>8</td>
<td>8</td>
<td>$2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-18.3</td>
<td>8</td>
<td>8</td>
<td>$2n^2$</td>
</tr>
<tr>
<td>8</td>
<td>1-SAPQ ($\eta = 4$)</td>
<td>Codebook Size (n')</td>
<td>2</td>
<td>32</td>
<td>128</td>
<td>$2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-6.31</td>
<td>32</td>
<td>64</td>
<td>$2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-12.8</td>
<td>32</td>
<td>64</td>
<td>$2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-18.8</td>
<td>32</td>
<td>64</td>
<td>$2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-23.9</td>
<td>2</td>
<td>16</td>
<td>$2n^2$</td>
</tr>
</tbody>
</table>

1 Number of multiplications per scalar value in encoding.
2 Number of scalar values for codewords.
3 m-SAPQ.

### Table 4.3: Distortion (dB) Comparison of SAPQ for Laplacian i.i.d. (Unit Variance) at k = 1

<table>
<thead>
<tr>
<th>Quan. Dim. (L)</th>
<th>Quantizer</th>
<th>$R = 1.5$</th>
<th>$R = 2.5$</th>
<th>$R = 3.5$</th>
<th>$R = 4.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>VQ</td>
<td>-6.30</td>
<td>-11.5</td>
<td>-17.2</td>
<td>-22.9</td>
</tr>
<tr>
<td>2</td>
<td>TSVQ (Breadth = 2)</td>
<td>-6.24</td>
<td>-11.0</td>
<td>-16.3</td>
<td>-22.1</td>
</tr>
<tr>
<td>2</td>
<td>1-SAPQ ($\eta = 1$)</td>
<td>-5.84 (-5.85)</td>
<td>-10.8</td>
<td>-16.3</td>
<td>-22.0</td>
</tr>
<tr>
<td>4</td>
<td>1-SAPQ ($\eta = 2$)</td>
<td>-6.31 (-6.38)</td>
<td>-11.5</td>
<td>-17.2</td>
<td>-23.0</td>
</tr>
<tr>
<td>8</td>
<td>1-SAPQ ($\eta = 4$)</td>
<td>-6.05 (-7.08)</td>
<td>-12.4</td>
<td>-18.1</td>
<td>-23.9</td>
</tr>
</tbody>
</table>

1 m-SAPQ.

### Table 4.4: Distortion (dB) Comparison between 1-SAPQ, PQ, L-dimensional VQ for i.i.d. Sources (Unit Variance) at $R = 0.5$

<table>
<thead>
<tr>
<th>Quan. Dim. (L)</th>
<th>Quantizer</th>
<th>Gaussian i.i.d.</th>
<th>Laplacian i.i.d.</th>
<th>Multiplications</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>VQ ($\nu = 16$)</td>
<td>-2.22</td>
<td>-2.69</td>
<td>16</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>TSVQ (Breadth=2, d = 4)</td>
<td>-1.68</td>
<td>-1.98</td>
<td>8</td>
<td>240</td>
</tr>
<tr>
<td></td>
<td>1-SAPQ ($k = 4, m = 2, \eta = 2$)</td>
<td>-2.20</td>
<td>-2.06</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>PQ ($k = 4, m = 2, n = 4$)</td>
<td>-1.89</td>
<td>-1.69</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>12</td>
<td>VQ ($\nu = 64$)</td>
<td>-2.32</td>
<td>-2.73</td>
<td>64</td>
<td>768</td>
</tr>
<tr>
<td></td>
<td>TSVQ (Breadth=2, d = 6)</td>
<td>-1.70</td>
<td>-2.01</td>
<td>12</td>
<td>1512</td>
</tr>
<tr>
<td></td>
<td>1-SAPQ ($k = 4, m = 3, \eta = 3$)</td>
<td>-2.24</td>
<td>-2.50</td>
<td>16</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>1-SAPQ ($k = 6, m = 2, \eta = 2$)</td>
<td>-2.27</td>
<td>-2.59</td>
<td>18</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>PQ ($k = 4, m = 3, n = 4$)</td>
<td>-1.89</td>
<td>-1.69</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>PQ ($k = 6, m = 2, n = 8$)</td>
<td>-2.07</td>
<td>-2.10</td>
<td>8</td>
<td>48</td>
</tr>
</tbody>
</table>
Figure 4.6: Comparison of SAPQ and VQ ($\eta = m/2$). (a) Number of codewords for codebook-constrained SAPQ and VQ. (b) Distortion (dB) of codebook-constrained SAPQ and VQ for a Gaussian i.i.d. with unit variance.
Figure 4.7: Distortions (dB) for various coding schemes for a Gaussian i.i.d. source with unit variance. (DRF: distortion rate function, TCQ Opt.: TCQ using optimized alphabets [34], TCQ: TCQ using rate-(R + 1) Lloyd-Max [34], Ent.Const.: entropy constraint optimum [25], Ent.Cod.: entropy-coded quantization, Lloyd-Max with Huffman [8], 1-SAPQ Ent.: 1-SAPQ with respect to its entropy. 1-SAPQ results are obtained by varying $\eta = 1, \cdots, 5$, for each $k$ and $n'$, and Trellis size in TCQ and quantizer dimension $km$ in 1-SAPQ are equal to 16.)
block decoder to reconstruct the quantized signal. TCQ can also be regarded as a structurally constrained VQ of length $km'$ where $m'$ is the searching depth in the trellis of TCQ [39]. In the 1-SAPQs of Fig. 4.7, the quantizer dimensions $L = km$ are equal to 16, and in the TCQ, $k = 1, m' = 1000$, and the number of state in the trellis is 16. We found that the 1-SAPQ distortions can be very close to that of TCQ when $L$ in 1-SAPQ and the trellis size is equal. For low bit-rates above 1 in Fig. 4.7, if $k = 1$, since the sample ratio is large as $\beta = m/n' = 4$, we cannot achieve a good gain from 1-SAPQ. However, if $k = 2$, then the sample ratio decreases to $\beta = 0.5$, hence we can obtain better gains than the $k = 1$ case. (In Fig. 4.7, only the $k = 2$ case is depicted above bit-rate 1. Note that the 1-SAPQ with $k = 1$ and $n' = 8$ has $\beta = 2$ and the 1-SAPQ with $k = 1$ and $n' = 16$ has $\beta = 1$.) The entropy coded quantization has variable-length outputs, hence the coding scheme suffers from the error propagation, data lost, and buffer control problems. TCQ also has an error propagation problem within the trellis size, and requires long search depth $m'$ in the Viterbi algorithm for achieving gains [54]. However, SAPQ has no error propagation property unless the side bits are corrupted. Even though the side bits are corrupted, the produced error is relatively small since the designed codebooks are very similar each other. This characteristic on SAPQ will be useful in developing robust quantization techniques for image and video coding in the error-resilient aspect. Further, unlike the long search depth in TCQ, by using a relatively short data, e.g., $L = 16$, we can obtain a large gain from SAPQ. By combining the entropy coders, SAPQ can achieve more gains, which are better than those of TCQ as we can see 1-SAPQ Ent. in Fig. 4.7. In Fig. 4.7, 1-SAPQ Ent. implies the 1-SAPQ distortions with respect to the entropy $H_{SAPQ}$ defined by

$$H_{SAPQ} := -\frac{1}{k} \sum_{j=1}^{2^k} \text{Pr}\{C_j \text{ used for } X\} \sum_{\ell=1}^{n'_{\ell}} P_{\ell\ell} \log_2 P_{\ell\ell} + \frac{\eta}{km'},$$

(4.6)

where $P_{\ell\ell} := (1/m) \sum_{i=1}^{m-1} \text{Pr}\{\ell \text{th codeword used for } X_i \mid C_j \text{ used for } X\}$. It will be necessary to design an entropy constrained SAPQ.

In a similar manner in the TCQ case for the Markov sources [25],[54], the proposed SAPQ can also be used for quantization of such Markov sources by combining the predictive coding scheme, which is based on scalar predictors. Furthermore, SAPQ can be easily applied to the quantization of non-independent and/or non-identically distributed signals by using different codebooks for each random vector $X_i$ as shown in $m$-SAPQ.
5. Differential Sample-Adaptive Product Quantizer

5.1 Introduction

Differential pulse code modulation (DPCM) is very widely used for speech and image data compression [39]. It is very easy to implement the DPCM coder, and the DPCM coder yields a short encoding delay, since DPCM considers only a small portion of the entire data when encoding. Thus, the DPCM coder can be easily adopted as part of a coding system due to its simplicity. The DPCM coders at 3 ~ 4 b/pixel can be used for broadcasting quality digitization of moving images at rates such as 34 Mb/sec ~ 45 Mb/sec. For example, we can find a simple DPCM coder in the JPEG (Joint Photographic Experts Group) algorithm. In JPEG, the DC coefficient is the average value of the 64 pixels of a DCT (discrete cosine transform) block. Since there exists a normally high correlation between the DC coefficients of adjacent DCT blocks, the quantized DC coefficient is encoded as the difference from the DC term of the previous block [82]. Suppose that \((x_0, x_1, x_2, \ldots)\) is the input sequence for the predictor in the DPCM coder and \(\hat{x}_t\) is a predicted value of \(x_t\). The prediction in JPEG can then be written as

\[
\hat{x}_t = x_{t-1}, \text{ for } t = 1, 2, \ldots.
\]

A similar DPCM coder is currently employed by the MPEG (Moving Picture Experts Group) algorithm. Note that, in JPEG and MPEG, both standards employ a uniform scalar quantizer, followed by an entropy coder or variable-length coder (VLC). However, VLC, such as the Huffman coder, suffers from error propagation, data loss, and buffer control problems. Recently, a significant problem in multimedia communication is the error-resilient transmission of image and video data over noisy communication channels, especially mobile wireless channels and networks with packet loss. Hence, the employment of VLC seems inappropriate to such erroneous environment [50],[10]. If we use the fixed-length coder (FLC) for a quantizer, instead of using VLC, the quantizer should be a Lloyd-Max (minimum mean square error) quantizer to minimize the reconstructed distortion. The ITU-R CMTT.721-1 is an example of such DPCM image coder at a high bit-rate. In either cases, in order to achieve the improved performance, we should employ the block source coding techniques, such as the vector quantizer (VQ), tree coder, or trellis coder, rather than the scalar quantizer [25]. However, since the performance and complexity of the DPCM coder is directly related to that of the prediction algorithm and the quantization scheme, introducing such block source coding techniques sacrifice the unique features of DPCM: fast encoding and short delay.

Let us consider adopting VQ in the DPCM coder. Let every \(m\) points of the input sequence be the vector input for a VQ, where the vector dimension or the block length is \(m\). Since VQ can exploit the intra block correlation [52], decorrelation techniques, such as DCT or DPCM, are not usually used to decorrelate the intra block correlation. By increasing the block length \(m\), we can improve the VQ gain. However, due to the encoding complexity, the block length is commonly limited to a size of 4 x 4 image blocks. Therefore, in highly correlated signals such as images, a high correlation exists among the neighboring blocks. This inter block correlation can be exploited by the predictive VQ (PVQ) scheme. Similar to the DPCM coder, PVQ predicts the current block from the previously encoded blocks, and the residual vectors, the difference between the original and the predicted vector, is then vector quantized. However, unlike the scalar predictor in DPCM, the prediction accuracy of vectors deteriorates because of the relatively long distance of pixels between blocks for the vector prediction [25]. The prediction is usually based on scalar predictors for each set of pixels [34],[20],[47]. Using the neural networks, a nonlinear vector predictor was also proposed [69]. Even though we can improve the coding performance by combining VQ and DPCM, the performance of PVQ is very dependent on the designed codebook. In order to exploit the varying statistical characteristics of images, it is preferred to employ a special technique, such as the universal source coding scheme, using the universal or super codebook [12],[85],[49],[63]. Furthermore, due to the high encoding complexity of VQ, if we design a low distortion (or high bit-rate) coder, then it is nearly impossible to design such coder by using a 4 x 4-dimensional VQ.
Next, we consider adopting the tree coder or the trellis coder in the DPCM coder. For a wide range of bit-rates, the tree and trellis coders can efficiently yield gains over the traditional DPCM coder that is based on scalar quantizers [25]. A 2-dimensional tree coder for images was proposed [58], where about 3 dB SNR improvement over DPCM was reported. Further in [34], the traditional scalar quantizer for the tree coder is replaced by a VQ to improve the gain. Besides the tree coder, the trellis coder or the trellis coded quantization (TCQ) technique can be employed in frame work of predictive coders [54]. Since the FLC-based TCQ is more robust than the VLC-based scalar quantizer and can obtain VQ-level performance, much research is recently being conducted, in order to design robust quantization schemes [50]. However, the tree and trellis coders have a relatively long search depth, yielding long encoding delay or requiring large data. Further, the tree and trellis coders have an error propagation problem, even though the error propagation of TCQ is limited within the trellis size.

In this section, we propose a simple DPCM coder by employing SAPQ. SAPQ simply uses multiple scalar quantizers, but, since SAPQ is a structurally constrained VQ, we can achieve VQ-comparable performance with a low encoding complexity even for high bit-rates. The FLC-based SAPQ has a robustness to the channel errors which is similar to that of scalar quantizers with FLC. The FLC-based SAPQ can achieve a comparable performance to TCQ, even for a short block length, such as 16. Furthermore, SAPQ can be combined with VLC. We can easily modify the traditional DPCM coder by replacing the scalar quantizer with SAPQ. Hence, in the proposed DPCM coder, we can use the same scalar predictor and adaptive techniques for the traditional DPCM coders [30]. The codebook for the proposed DPCM coder is predesigned for given synthetic data, and can be scaled, according to the input variance in a similar way of the traditional DPCM coder. Since the proposed DPCM coder is based on a short block length, yielding a short encoding delay, and it is easy to combine the proposed scheme with the other coding schemes. In this section, we will refer to this scheme as predictive SAPQ or differential SAPQ (DSAPQ). By experiment, it is shown that more than 4 dB SNR improvement is obtained from DSAPQ with FLC.

This section is organized as follows. In Section 5.2, the proposed DSAPQ is introduced, by adopting SAPQ in the DPCM coder. In Section 5.3, we describe the design of DSAPQ using the training sequence (TS), and simulation results, together with discussions, are provided in Section 5.4. We then conclude the section in the last section.

5.2 Differential Sample-Adaptive Product Quantizer

In order to describe the proposed DSAPQ, we consider \( Lm + 1 \) random variables \( X_0, X_1, \ldots, X_{Lm} \), where \( L \in \mathbb{N} \). Let an observation of \( X_0, X_1, \ldots, X_{Lm} \) be \( x_0, x_1, \ldots, x_{Lm} \). We can then define the error signal \( \delta_t \) as

\[
\delta_t := X_t - \hat{X}_t,
\]

which is a random variable defined on the same underlying sample space \( \Omega \). For a given codebook size \( n \), if we use a codebook \( C \in \mathbb{C}^n \) for the quantization of all the error signal \( \delta_t \), then the average distortion of the conventional DPCM coder can be expressed as

\[
D_{\text{DPCM}} := E \left\{ \frac{1}{Lm} \sum_{t=1}^{Lm} \min_{y \in C} (\delta_t - y)^2 \right\},
\]

where we assume that the random variables \( \delta_t \) are identically distributed. Let \( y_t \in C \) denote the quantized value of \( \delta_t^w = x_t - \hat{x}_t \), where \( w \) is a sample point in \( \Omega \). We can then express the reconstructed values \( z_t \) as

\[
z_t := y_t + \hat{x}_t.
\]
Note that the average distortion in (5.3) can be rewritten as

\[ D_{\text{DPCM}} = E \left\{ \frac{1}{Lm} \sum_{t=1}^{Lm} \min_{y \in C} (X_t - Z_t)^2 \right\}, \]  \hspace{1cm} (5.5)

where \( X_t \) and \( Z_t \) are the input and reconstructed random variables, respectively.

In order to compare the performance of DSAPQ, we now examine an example of the conventional DPCM coder. Consider identically distributed random variables with variance \( \sigma^2_X \). Assuming that the DPCM coder employs a first-order linear predictor, then the optimal prediction coefficient \( a_1 \), which minimizes the average error

\[ E \left\{ \frac{1}{Lm} \sum_{t=1}^{Lm} (X_t - a_1X_{t-1})^2 \right\}, \]  \hspace{1cm} (5.6)

is \( a_1 = p \), and the variance of the error signal \( X_t - a_1X_{t-1} \) is

\[ \text{Var} \{X_t - a_1X_{t-1}\} = (1 - \rho^2)\sigma^2_X, \]  \hspace{1cm} (5.7)

where \( p \) is the correlation coefficient. In order that the transmitter and receiver track and reconstruct input sequence in synchrony, the prediction of \( \hat{Z}_t \) should be a function of the previously reconstructed values of \( z_{t-1}, z_{t-2}, \ldots \). Hence, in the DPCM coder, we normally consider a predictor:

\[ \hat{Z}_t = a_1z_{t-1}. \]  \hspace{1cm} (5.8)

If the quantization noise is very small, then we can approximate the relation in (5.8) to \( a_1z_{t-1} \). (For example, for \( R > 2 \), we can make the approximation [39, p.260].) Hence, the error signal \( \delta_t (\approx X_t - a_1X_{t-1}) \) has the variance in (5.7) approximately. In (5.7), the quantity \( (1 - \rho^2) \) is always less than 1, if \( |\rho| < 1 \). Hence, if there is a correlation between the adjacent random variables, then we can expect a gain from DPCM. As an example for a Markov-1 source, we could reduce the distortion by \(-10 \log(1 - \rho^2)\) dB, since the quantizer distortion is proportional to the input variance. However, since the prediction is based on the quantized values, the gain will be less than \(-10 \log(1 - \rho^2)\) dB. For a Gaussian Markov-1 source with the correlation coefficient 0.9, we see numerically 7.15 dB improvement for the codebook size \( n = 16 \), which is very close to the theoretical gain 7.21 dB. Note that the codebook \( C \), which is used in (5.3), is the optimal Lloyd-Max codebook for the error signal \( X_t - a_1X_{t-1} \) [56]. (The error signal \( X_t - a_1X_{t-1} \) has the Gaussian i.i.d. pdf with variance \( (1 - \rho^2)\sigma^2_X \)).

We now describe the proposed DSAPQ, based on the DPCM coder. By introducing the indexes, \( i \) and \( h \), define an error signal \( \delta_i(h) := \delta_{m(h-1)+h-1} \) for \( i = 1, \ldots, m \) and \( h = 1, \ldots, L \). The average distortion of the DPCM coder of (5.3) can then be rewritten as

\[ D_{\text{DSAPQ}} := E \left\{ \frac{1}{L} \sum_{h=1}^{L} \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C} (\delta_i(h) - y)^2 \right\}. \]  \hspace{1cm} (5.10)

We now replace the quantization part of (5.9), where only one codebook \( C \) is used, with a SAPQ, employing \( N(=27) \) different codebooks, \( C_1, \ldots, C_N \). Here, for a given codebook size \( n' \), the codebooks belong to \( C_{n'} \). The average distortion of DSAPQ is then given by

\[ D_{\text{DSAPQ}} := E \left\{ \frac{1}{L} \sum_{h=1}^{L} \min_{j \in \{1, \ldots, N\}} \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_j} (\delta_i(h) - y)^2 \right\}. \]  \hspace{1cm} (5.10)

In this quantizer, from (4.3), the bit rate is given by

\[ R = \log \frac{n' + \eta}{m}. \]  \hspace{1cm} (5.11)
As shown in (5.10), DSAPQ is an SAPQ for the input \( \delta_t(h) \). Hence, in DSAPQ, we can use \( mN \) different codebooks \( C_{i,j} \), depending on the index \( i \) in a similar manner of the SAPQ, we have a relationship:

\[
\inf_{DSAPQ} D_{\text{DSAPQ}} \leq \inf_{D\text{PCM}} D_{\text{DPCM}}
\]  

(5.12)

under the assumption that \( \delta_t \approx X_t - a_1X_{t-1} \). However, if \( m \) is large, then the decoder requires a large memory for the \( mN \) codebooks, implying that the codebook design complexity is high. In order to reduce the required number of codebooks, DSAPQ uses a codebook \( C_j \) for calculating the distance of \( m^{-1} \sum_{i=1}^{m} \min_{y \in C_j} (\delta_t(h) - y)^2 \) for a fixed \( j \). In other words, \( C_{i,j} \) are set equal to \( C_j \) for \( i = 1, \ldots, m \). Therefore, we need only \( N \) codebooks in \( C_j \). However, we cannot always guarantee the relationship in (5.12). The employed SAPQ in the DSAPQ of (5.10) is a codebook-constrained SAPQ, since \( C_{i,j} \) are the same for all \( i \).

We now examine the performance of the codebook-constrained SAPQ. As mentioned previously in Section 2.2, increasing \( m \) for large \( n' \) yields more gain over PQ in the SAPQ case. However, for the codebook-constrained SAPQ case, the decrease in distortion can be seen to diminish for large \( m \), and the distortion will eventually increase and converge to that of the \( n' \)-level quantizer \([45]\). In other words, even though we increase the side bit \( \eta \) in (5.11), if the ratio \( m/n' \) is high, then we cannot expect any gain from increasing \( \eta \). Therefore, to obtain gains in the codebook-constrained SAPQ, it is important to use a large value of \( m \) (and \( n' \)), while keeping the ratio \( m/n' \) small as a "design guideline". This design guideline is also effective to the DSAPQ case. Numerical results and discussion on the performance of the codebook-constrained SAPQ and DSAPQ will be presented in Section 5.3.

A block diagram for the DSAPQ is shown in Figs. 5.1(a) and (b). The SAPQ is illustrated in Fig. 5.1(a) and the DPCM loop, \( DQ_j \), is illustrated in Fig. 5.1(b), respectively. In Fig. 5.1(a), the total number of choices of the codebooks or DPCM coders is \( N = 2^n \). In order that the transmitter and receiver track and reconstruct the input sequence in synchrony, the prediction depends on previously quantized values, yielding the feedback-loop shown in Fig. 5.1(b). Hence, the basic operation is the same as the scalar quantizer cases. However, as mentioned previously, the improved gain can be achieved, which is summarized in Section 2.3. In other words, we achieve VQ-level performance.

### 5.3 DSAPQ Design Based on Clustering

The SAPQ codebook for DSAPQ should be optimal for the error signal \( \delta_t(h) = X_t - \tilde{X}_t \). But, because of the quantization error, it is difficult to clarify the characteristics of the error signal. However, we can approximate \( \delta_t(h) = X_t - \rho X_{t-1} \) to \( \delta_t(h) \approx X_t - \rho X_{t-1} \), assuming that the quantization error is very small. If the input is the Gaussian Markov-1 source, then the pdf of the error signal \( \delta_t(h) \) is approximately i.i.d. Gaussian with variance \( (1 - \rho^2)\sigma_X^2 \). Hence, designing DSAPQ for the Gaussian Markov-1 source is equivalent to designing the SAPQ codebook for the Gaussian i.i.d. source with variance \( (1 - \rho^2)\sigma_X^2 \). Based on this notion, we design the SAPQ codebook, and then use the codebook for the DSAPQ, by scaling the designed SAPQ codebook accordingly.

In Fig. 5.2(a), the designed codebooks for the codebook-constrained SAPQ and the PQ in (2.2) are compared for a Gaussian i.i.d. source with variance \( \sigma_X^2 \). The quantizers used for the PQ are the Lloyd-Max quantizers \([56],[65]\). For example, for \( m = 16 \) and \( \eta = 4 \) in Fig. 5.2(a), the distortion of the codebook-constrained SAPQ shows an 1.8 dB improvement over PQ. We use the designed codebooks for DSAPQ by scaling the variance and the mean, and the numerical results are shown in Fig. 5.2(b). For the \( m = 8 \) case in Fig. 5.2(b), the gains are slightly less than the PQ and SAPQ cases. However, the other cases, it is noted that the gains are nearly the same as shown in Fig. 5.2(a). In a similar manner, we illustrate the numerical results for the Laplacian i.i.d. and the Laplacian Markov-1 sources in Figs. 5.3(a) and (b), respectively. Figs. 5.3(a) and (b) show the results that are consistent with the previous discussion in Section 2.2, regarding that the Laplacian source yields more gains. As shown in Fig. 5.3(a), the distortion of the codebook-constrained SAPQ shows an 2.9 dB improvement over PQ for \( m = 16 \) and \( \eta = 4 \).
Figure 5.1: Block diagram of DSAPQ. (a) SAPQ part \((N = 2^n)\). (b) DQ: the \(j\)th DPCM part with a distortion accumulator for DSAPQ \((j = 1, \ldots, N)\).
Figure 5.2: The codebook-constrained SAPQ and DSAPQ for the Gaussian pdf \((n' = 16)\). (a) Codebook-constrained SAPQ designed for Gaussian i.i.d. with \(\sigma_X^2 = 1\). (b) DSAPQ for Gaussian Markov-1 with \(\sigma_X^2 = 1\) and \(\rho = 0.9\) \((24 = \rho_{z_{t-1}})\).

Figure 5.3: The codebook-constrained SAPQ and DSAPQ for the Laplacian pdf \((n' = 16)\). (a) Codebook-constrained SAPQ designed for Laplacian i.i.d. with \(\sigma_X^2 = 1\). (b) DSAPQ for Laplacian Markov-1 with \(\sigma_X^2 = 1\) and \(\rho = 0.9\) \((24 = \rho_{z_{t-1}})\).
In this section, it is assumed that \( \log_2 n_i \) and \( \log_2 n_i' \) could be non-integer for performance comparison purpose. Note that, in these simulations, we maintain the training ratio, the size of TS to the total number of codewords, such that the ratio is higher than 5,000 to ensure a good codebook for the underlying distribution function of the TS [41]. For example, in the case of \( m = 16 \) and \( \eta = 4 \), the total number of codewords is 256. Hence, the TS size should be greater than 1,280,000. However! in validating the trained codebook for the DPCM and DSAPQ, we do not need such a large size of VS [42].

5.4 Numerical Results

In order to examine the performance improvement, in terms of the distortion and the subject quality, first, a zig-zag scanned image is quantized using DSAPQ. In the first simulation, the 2-dimensional image is zig-zag scanned into an 1-dimensional signal, and we use a simple predictor, given by

\[
\hat{x}_t = z_{t-1}.
\]

(5.13)

Note that the variance of the error signal \( X_t - X_{t-1} \) is \( \text{Var}\{X_t - X_{t-1}\} = 2(1 - \rho)\sigma_X^2 \). Hence we have

\[
\text{Var}\{X_t - Z_{t-1}\} \approx 2(1 - \rho)\sigma_X^2.
\]

(5.14)

In Table 5.1, the estimated means, variances, and the correlation coefficients for two different real images are listed. In the following tables, \( \sigma_X^2 \) implies the variance of the error signal, where the predictor uses the non-quantized data, i.e., \( z_{t-1}, z_{t-2}, \cdots \). Hence, in Table 5.1, \( \sigma_X^2 \) implies the estimate of \( \text{Var}\{X_t - X_{t-1}\} \). From Table 3, we note that the error signals yields much lower variances than the original image signal, \( \sigma_X^2 \). The PSNRs (peak signal to noise ratios) of DPCM and DSAPQ are summarized in Table 5.2, by varying the adaptation periods \( m \). In Table 5.2, we consider two different quantizers for the Gaussian and Laplacian pdfs, respectively, as described in Section 5.3. We note that the distribution of the error signal of the image is close to the Laplacian pdf, rather than the Gaussian pdf, and DSAPQ yields the gains about \( 2 \sim 3 \) dB over the DPCM coder. In Table 5.2, the PSNRs for Bridge are better than those of Lena, even though the error signal variance of Lena is smaller than Bridge as shown in Table 5.1. It is believed that this observation is due to the mismatched pdf shapes [39]. (The variance is matched by

| Table 5.1: Variances and Correlation Coefficients of Real Images |
|---------------------|-----|-----|-----|-----|
| Image   | \( \rho \) | Mean | \( \sigma_X^2 \) | \( \hat{x}_t = \rho z_{t-1} \) | \( \hat{x}_t = \sigma_X^2 \) |
| Lena    | 0.9719 | 124.1 | 2290 | 126.8 | 128.6 |
| Bridge  | 0.9403 | 113.8 | 2996 | 347.0 | 357.6 |

| Table 5.2: PSNRs (dB) of DPCM and DSAPQ for Zig-Zag Scanned Images and \( \hat{x}_t = z_{t-1} \) Quantizers are Scaled by the Corresponding Standard Deviations \( \sigma_X \) |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| Image | Quantizer | DPCM (\( n = 19 \)) | DSAPQ (\( n' = 16, R = 4.25 \) b/pixel) |
| Lena   | Laplacian Gaussian | 40.50 | 40.63 | 41.95 | 42.80 | 43.95 |
|        |                  | 34.45 | 34.10 | 34.77 | 35.30 | 36.37 |
| Bridge | Laplacian Gaussian | 41.84 | 42.41 | 43.30 | 43.43 | 44.04 |
|        |                  | 37.13 | 37.80 | 38.37 | 38.68 | 39.71 |
scaling using the corresponding variance.) If we scale the variance using other values, which are different from the variance of the error signal, then we can obtain better performance for the DPCM and DSAPQ coders, especially for Lena; numerical results are shown in Table 5.3.

<table>
<thead>
<tr>
<th>Image</th>
<th>Scale Factor</th>
<th>( \sigma_{\xi} )</th>
<th>DPCM ((n = 19, R = 4.25 \text{ b/pixel}))</th>
<th>DSAPQ ((n' = 16, R = 4.25 \text{ b/pixel}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>18</td>
<td>111.341</td>
<td>44.35</td>
<td>44.72</td>
</tr>
<tr>
<td>Bridge</td>
<td>21</td>
<td>18.91</td>
<td>41.89</td>
<td>42.59</td>
</tr>
</tbody>
</table>

Table 5.3: PSNRs (dB) of DPCM and DSAPQ
FOR ZIG-ZAG SCANNED IMAGES AND \( \hat{z}_t = z_{t-1} \)
LAPLACIAN QUANTIZERS ARE SCALED BY THE SCALE FACTORS

The DPCM coded images usually suffer from the granularity and the slope-overload distortion. In Fig. 5.4, magnified portions of the reconstructed images are illustrated to show the slope-overload distortion among the several quantization results in Table 5.2. In the traditional DPCM coder of Fig. 5.4(b), we note that the zig-zag shaped artifact at the edge of the "hat". This artifact arises from the zig-zag scanning and the slope-overload distortion following the scanning direction. However, we can see that the artifact is significantly alleviated in the DSAPQ case (see Figs. 5.4(c) and (d)).

We next employ a 2-dimensional, second order predictor for DSAPQ, as shown in Fig. 5.5. This predictor is also used in ITU-R CMTT.721-1. The numerical results for various images are presented in Tables 5.4 and 5.5, respectively. In these tables, we also use the codebook-constrained SAPQ codebooks that are designed for the Laplacian pdf for different parameters of \( \eta \) and \( \sigma_{\xi} \). In the 2-dimensional predictor case, we can also obtain similar gains from DSAPQ. Figs. 5.6(a) and (b) illustrate the PSNRs of DPCM and DSAPQ at various bit-rates.

The proposed DSAPQ for image compression is very similar to an adaptive DPCM image coder introduced in [29, 374], where four differently scaled versions of the Lloyd-Max quantizers are employed. However, it should be noted that the adaptive DPCM coder exploit the varying variance of signals, which is quite different from the notion of sample adaptation in DSAPQ. We can also design an adaptive DSAPQ, by exploiting the varying statistical characteristics.

Table 5.4: PSNRs (dB) of DPCM and DSAPQ
2-DIMENSIONAL PREDICTOR: \( X = (A + B)/2 \)
LAPLACIAN QUANTIZERS SCALED BY THE CORRESPONDING STANDARD DEVIATIONS \( \sigma_{\xi} \)

<table>
<thead>
<tr>
<th>Image</th>
<th>( \sigma_{\xi} )</th>
<th>DPCM ((n = 19, R = 4.25 \text{ b/pixel}))</th>
<th>DSAPQ ((n' = 16, R = 4.25 \text{ b/pixel}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>7.918</td>
<td>41.23</td>
<td>40.45</td>
</tr>
<tr>
<td>Bridge</td>
<td>15.71</td>
<td>43.72</td>
<td>44.60</td>
</tr>
</tbody>
</table>

5.5 Discussions on the Generalized SAPQ and DSAPQ

As shown in Fig. 4.1, for a correlated source such as the Gauss-Markov source, we can achieve a large gain from 1-SAPQ at \( k > 1 \), i.e., from the generalized SAPQ. However, the maximum gain in Fig. 4.1 is about 6 dB over the Lloyd-Max quantizer. As shown in Figs. 5.2(a) and (b), we can easily obtain more than 7 dB gain from the traditional DPCM with Lloyd-Max quantizer for the Markov-1 source. However, this situation does not always hold for the general signals. We now compare the 1-SAPQ performance with those of DPCM and DSAPQ for real images. From experimental results, the performance of 1-SAPQ...
Figure 5.4: Parts of the DPCM and DSAPQ coded images of Lena. (a) The original image. (b) DPCM with $n = 19$ ($R \approx 4.25\text{ b/pixel}, 40.50\text{ dB}$). (c) DSAPQ with $n' = 16, \eta = 2$, and $m = 8$ ($R = 4.25\text{ b/pixel}, 41.95\text{ dB}$). (d) DSAPQ with $n' = 16, \eta = 4$, and $m = 16$ ($R = 4.25\text{ b/pixel}, 43.95\text{ dB}$).
Figure 5.5: 2-dimensional predictor for image data: $X = (A + B)/2$.

Figure 5.6: PSNR comparison between DPCM and DSAPQ for real images with respect to $n'$ (2-dimensional predictor: $X = (A + B)/2$, $m = 16$, and $\eta = 4$). (a) Lena (Scale factor = 14). (b) Bridge (Scale factor = 16).
Table 5.5: PSNRs (dB) of DPCM and DSAPQ
2-DIMENSIONAL PREDICTOR: \( X = (A + B)/2 \)

<table>
<thead>
<tr>
<th>Image</th>
<th>Scale Factor</th>
<th>DPCM (( n = 19 )) ( R \geq 4.25 \text{ b/pixel} )</th>
<th>DSAPQ (( n' = 16, R = 4.25 \text{ b/pixel} ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>14 (7.918)</td>
<td>47.02</td>
<td>47.14</td>
</tr>
<tr>
<td>Bridge</td>
<td>16 (16.71)</td>
<td>43.71</td>
<td>44.60</td>
</tr>
</tbody>
</table>

and DSAPQ with the 2-dimensional predictor show very similar distortions, as shown in Figs. 5.7 and 5.8. This means 1-SAPQ is better than the traditional DPCM. (In fact as shown in Figs. 5.4 and 5.9, the 1-SAPQ of Lena at 1 b/pixel is 31.8 dB, but the DPCM PSNR is 25.8 dB.) However, the visual quality of DSAPQ is better than that of 1-SAPQ, because of the blocking artifacts of VQ. On the other hand, 1-SAPQ is robust to the noise, but DSAPQ is not. However, the error-resilience of DSAPQ is better than a combination of the traditional DPCM and VLC. While conducting the research, we should find a good quantizer scheme in terms of distortion and error-resilience.

5.6 Conclusion

In this section, we considered a new quantization scheme, SAPQ, to quantize image data within a framework of the DPCM coder. The employed SAPQ is a structurally constrained VQ, based on the scalar quantization to reduce the encoding complexity of the VQ. SAPQ has a form of the feed-forward adaptive quantizer with a short adaptation period. However, since SAPQ is a structurally constrained VQ, SAPQ can achieve VQ-level performance, even at high bit rates, while requiring the comparable complexity to the scalar quantizers. Since SAPQ has a scalar quantizer structure, we can easily apply SAPQ to the existing DPCM coder, in order to obtain VQ-level performance. Depending on the input data and bit-rate, we could obtain more than 4 dB gain over the traditional DPCM coder at the block length of 16. This gain is comparable to the gains from the tree or trellis coders for large block length. Furthermore, by combining the entropy coders we can expect more gains from the proposed DSAPQ.
Figure 5.7: Reconstructed images at 1.25 b/pixel (Lena). (a) $k = 2 \times 2$, $m = 2 \times 2$, $n' = 16$, and $\eta = 16$ 1-SAPQ (33.0 dB). (b) $k = 1$, $m = 16$, $n' = 2$, and $\eta = 16$ DSAPQ with 2-dimensional predictor (32.7 dB).

Figure 5.8: Reconstructed images at 1.25 b/pixel (Bridge). (a) $k = 2 \times 2$, $m = 2 \times 2$, $n' = 16$, and $\eta = 16$ 1-SAPQ (27.9 dB). (b) $k = 1$, $m = 16$, $n' = 2$, and $\eta = 16$ DSAPQ with 2-dimensional predictor (28.5 dB).
Figure 5.9: DPCM with the 2-dimensional predictor at 1 b/pixel. (a) Lena (25.8 dB for scale factor = 14). (b) Bridge (23.3 dB for scale factor = 16).

Figure 5.10: DPCM with the 2-dimensional predictor at 2 b/pixel. (a) Lena (33.0 dB for scale factor = 14). (b) Bridge (30.1 dB for scale factor = 16).
6. Robustness of SAPQ in Channel Noise

6.1 Introduction

In recent years, the error-resilient transmission of image and video data over noisy channels become an important problem, especially due to the rapidly growing demand of multimedia and wireless services. Hence, developing robust quantization schemes by combining the source and channel coder design problems is important. For example, in the scalar quantizer (SQ) case, protecting only the most significant bit (MSB) can dramatically reduce the channel noise effect on the quantizer performance, since only one bit difference in MSB could yield large errors in quantizers.

There is a significant body of research in the area of channel coding that is devoted to studying the problem of robustness to channel noise. However, there is very little work on developing a quantization scheme itself that is both robust to channel noise and yields good performance. In this section, we will focus on the robustness of SAPQ, for the binary symmetric channel (BSC).

There is related work on how to assign binary codes to the output levels of a quantizer in order to reduce the effect of channel noise on the system [72] (this work assumes a given quantizer, i.e., it does not explicitly take into account the quantization scheme). The natural binary code (NBC) and the folded binary code (FBC) are usually employed for the outputs of SQs, such as the Lloyd-Max quantizers. By performing a judicious assignment of binary codes to index the codewords, we can alleviate the distortion due to the channel noise. A binary code assignment constructed by such assignment is called the minimum distance code (MDC) for a given source and quantizer. A significant body of research has been conducted on finding the MDC for SQ and VQ in [72], [53], [17], and [35].

In other related work, combining the source and channel coding scheme has been studied to take the channel into consideration when designing the quantizer, even when the system is not constructed to operate in real time [48], [19]. The combined source-channel coder can be accomplished by arranging the quantizer such that codewords having a small Hamming distance in terms of the channel code also have reconstruction points with small signal distances. A mapping of a binary block code to generate the reconstruction vectors of VQ was proposed in [31]. Joint source and channel trellis waveform coders [3] and channel optimized trellis coded quantization (TCQ) techniques have [83] also been developed.

In both classes of schemes described above, the quantizer is not explicitly considered, i.e., it is used as a "black box." However, in our work, we explicitly consider designing a robust quantizer, hence our scheme could be combined with any of these schemes to achieve greater system performance.

It is known that, when the coded quantizer output is transmitted via a very noisy channel, quantizers with a small number of levels yield better performance than those having a large number of levels [76], [39]. In order to improve quantization gains, employing VQ techniques have received tremendous attention in the field of data compression. An efficient data compression scheme removes the redundancy from the source to the extent possible. However, this removal of redundancy, in turn, can produce a great deal of sensitivity to channel noise [17]. In general, VQ is more sensitive than SQ, and it is more difficult to appropriately assign the binary codes to the codewords of VQ than the SQ case. The sensitivity to channel noise can be compared by plotting the quantized and corrupted distortion with respect to various bit error rates (BERs), and observing the classical distortion bound due to the channel noise [39]. This bound is called the channel-limit performance [39]. As the vector dimension of VQ increases, the channel-limit performance deteriorates. If we use a SQ followed by the variable-length coding (VLC) scheme, then, since VLC suffers from error propagation and synchronization problems due to data loss, it seems that employing VLC is extremely inappropriate for such an error-filled environment. Note that, in the VLC-based coding scheme, a one bit error alone could produce a fatal degradation in reconstruction. For high bit-rates, TCQ can be used to improve the quantization performance, since TCQ is relatively insensitive to the channel noise [54], [50]. However, TCQ has a relatively long search depth and also an error propagation problem within the trellis size. Therefore, TCQ is also more sensitive than SQ to the channel noise.

In this section, we will demonstrate through numerical results that the proposed quantization scheme
has better channel-limit performance than the full search VQ, and provide several examples with vector quantization of real images to further illustrate this point.

This section is organized as follows. In Section 6.2, the channel-limit performance of SQ and VQ are demonstrated. For synthetic data, the channel noise effect on SAPQ is then observed, by comparing to the other quantizers, in Section 6.3. We quantize the real images based on VQ to show the performance of the proposed quantizer and the channel noise effect in Section 6.4. We then conclude the section in the last section.

6.2 Channel-Limit Performance

In this section we consider the effect of a noisy channel on SAPQ. We assume that random errors are introduced in the bits that convey information about quantizer outputs over a binary symmetric channel. We also assume that the channel is a stationary memoryless channel.

When we encode each quantizer level, based on the fixed-length coding framework, we generally consider NBC and FBC for representing the quantizer levels \[39\]. The binary codes in NBC are the binary representations of the index of codewords 1 to \(2^R\). For example, if a binary code is given by \(b_1 b_2 \ldots b_R\), then the corresponding quantizer codeword \(y_i\) has the index of \(i = 1 + \sum_{r=1}^{R} 2^{R-r} b_r\). For the Lloyd-Max quantizers, MSB, \(b_1\), is a sign bit. The remaining \((R-1)\) bits are used to transmit magnitude information. For the Laplacian pdf, the FBC is identical to the MDC for \(R = 2\) and 3 \[72\]. In this section, in all our comparisons (numerical and with real images) on the effect of channel noise on SAPQ, VQ, etc., we use the well known NBC and FBC schemes for assigning quantizer levels.

In Fig. 6.1(a), the quantizer performance of the Lloyd-Max quantizers over several noisy channels, having different bit error rates (BERs), is illustrated in terms of the reconstructed distortion. It is shown that, for a bit-rate of \(R = 4\), FBC is significantly better than NBC. Also in Fig. 6.1(a), it can be seen

\[\text{Figure 6.1: Distortion (dB) of quantizers with respect to bit error rate, and channel-limit performance. (a) Scalar quantizers (m = 1) for various bit rate (NBC: natural binary code, FBC: folded binary code). (b) Channel-limit performance, C1, C2, and C3 for SQ \((k' = 1)\), VQ \((k' = 2)\), and VQ \((k' = 4)\), respectively.}\]
that at high bit-rates, the quantizer system is more sensitive to increasing BER (especially at high BERs, an observation also made in [39]). Note that the curves for FBC seem to converge to a particular level of distortion as the BER increases, since the distortions are bounded by the same channel limit performance [38]. In Fig. 6.1(b), the channel-limit performance for different vector dimensions of VQs are compared with that of SQ. Note that the VQ in Fig. 6.1(b) was designed by using GLA and the "splitting" algorithm for the initial guess. It is known that the NBC assignment based on the splitting algorithm performs very well for the VQ case [17]. Hence, we use NBC for the simulation of VQ. Based on these simple assignments, we can see that SQ has the best channel-limit performance as shown in Fig. 6.1(b), even though the quantizer distortions of the VQs are better than that of the SQ, when there is no channel noise.

6.3 Error Performance of SAPQ

We now observe the channel noise effect on the proposed quantization scheme, SAPQ. Note that the bit stream of SAPQ is composed of two types: the main bits for the codeword index and the side bits for the codebook index, respectively. Consider the $k = 1$ case, then each codebook $C_j$, which is used for the SAPQ in (4.5), is also a subset of $R$. Hence for the main bits, since the SAPQ operation is the same as that of SQ, we can design a MDC for the corresponding codebooks based on SQ or simply use BNC or FBC. For both the Gaussian and Laplacian pdfs, as in SQ, FBC performs better than PBC, in the SAPQ case as well. Further, as in the fixed-length code based SQ, SAPQ has no error propagation property (since in the $k = 1$ case, SAPQ is based on a SQ structure). If the side bits corresponding to a sample (or m input vectors) in SAPQ are corrupted, then a codebook that results in a higher distortion may be chosen for that particular sample. However, even if the side bits were corrupted, the produced error is relatively small, since the codewords of designed codebooks are very similar to each other at the same codeword index as shown in Table 6.1'. Therefore, we can expect that the channel-limit performance of SAPQ, which is based on SQ ($k = 1$), is similar to that of the conventional SQ. This fact will also be demonstrated in the following numerical results.

<table>
<thead>
<tr>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
<th>$C_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_{1,j}$: -2.80</td>
<td>-1.84</td>
<td>-2.19</td>
<td>-1.71</td>
</tr>
<tr>
<td>$y_{2,j}$: -1.49</td>
<td>-1.26</td>
<td>-1.21</td>
<td>-1.02</td>
</tr>
<tr>
<td>$y_{3,j}$: -0.88</td>
<td>-0.77</td>
<td>-0.64</td>
<td>-0.52</td>
</tr>
<tr>
<td>$y_{4,j}$: -0.38</td>
<td>-0.30</td>
<td>-0.16</td>
<td>-0.07</td>
</tr>
<tr>
<td>$y_{5,j}$: 0.07</td>
<td>0.16</td>
<td>0.29</td>
<td>0.39</td>
</tr>
<tr>
<td>$y_{6,j}$: 0.52</td>
<td>0.66</td>
<td>0.73</td>
<td>0.88</td>
</tr>
<tr>
<td>$y_{7,j}$: 1.02</td>
<td>1.28</td>
<td>1.18</td>
<td>1.47</td>
</tr>
<tr>
<td>$y_{8,j}$: 1.73</td>
<td>2.19</td>
<td>1.79</td>
<td>2.79</td>
</tr>
</tbody>
</table>

In Fig. 6.2, the performance of SAPQ is compared to different quantization schemes in order to observe its channel-limit performance. In this figure, "SQ+Huffman" is the 8-level Lloyd-Max quantizer with the Huffman code. The entropy coded quantization or the variable-length coder (VLC), such as the Huffman code, has variable-length outputs, hence the coding scheme suffers from serious data synchronization problems due to the error propagation and data loss. Hence, even for very low values of the BER, the channel noise effect is extremely bad, as shown in Fig. 6.2. Thus, employing VLC-based quantization in

---

Note: The codewords of $C_1$ and $C_4$, and $C_2$ and $C_3$ are approximately symmetric with respect to the origin. By exploiting this property, we could reduce the size of memory required for codebooks; but this is another research topic.
Figure 6.2: Different quantization schemes and channel-limit performance, C1 for SQ \( (k' = 1) \) and SAPQ \( (k = 1, m/n' = 4) \), and C2 for VQ \( (k' = 2) \).

an error-filled environment seems inappropriate. In the current image and video coding schemes, several synchronizing bits, which are composed of consecutive long zeros, are employed to alleviate the effect of error propagation and data loss [82]. However, the data between the synchronization bits, suffers from error propagation and data loss problems, when using VLC-based coding schemes. In Fig. 6.3, the SAPQ and VQ of Table 4.2 are compared at the same bit-rates. In this case, SAPQ can obtain even better performance than VQ, while maintaining a better channel-limit performance (C1 for SAPQ versus C2 for VQ).

For the \( k = 1 \) (scalar) case, since the structure of SAPQ is that of a scalar quantizer, if we can protect (e.g., applying channel error correction to) only a few important bits, such as the MSB from the channel noise, then we can obtain improved performance. The SAPQ performance with respect to various number of protected bits is compared with that of VQ in Fig. 6.4(a). The channel-limit performance of the MSB protected SAPQ is also illustrated in Fig. 6.4(b) compared to the VQ. In the SAPQ case, by only protecting the MSB, we can provide more than 3 dB improvement over SAPQ without MSB protection. However, in the VQ case as shown in Figs. 6.4(a) and (b), improving the performance of VQ by simply protecting a few bits seems difficult.

6.4 Image Vector Quantization

We now observe the channel noise effect on the reconstructed images. In this simulation, we use NBC for the binary bit assignment of VQ. In Fig. 6.5, we illustrate the channel noise effect on the quantized images in Fig. 4.5, respectively. We can see the "salt-and-pepper" noise due to the channel noise in both the VQ and SAPQ. However, the SAPQ continues to show better quality over the VQ even for the highly corrupted images. The channel-limit performance for the images in Fig. 6.5 for different BER, are shown in Fig. 6.6. Note that, since the SAPQ is based on VQ with \( k = 4 \), the channel-limit performance of the SAPQ is also similar to that of the VQ having \( k' = 4 \) as shown in Fig. 6.6.

We now compare SAPQ with VQ based on VLC (so that the VLC enhances the performance of the VQ). In order to improve the coding gain, we increase the codebook size of VQ, \( \nu' \), to 32 and use the
Figure 6.3: Comparison of SAPQ ($k = 1$) and VQ ($k' = 2$) for the Gaussian i.i.d. source (C1: SAPQ, C2: VQ). (a) SAPQ: $m = 2$ and $\eta = 1$. (b) SAPQ: $m = 8$ and $\eta = 4$.

Figure 6.4: Bit protection of SAPQ ($k = 1, m = 8, n' = 16, \eta = 4$) and VQ ($k' = 2, \nu' = 512$) for the Gaussian i.i.d. source at $R = 4.5$. (a) Distortion with respect to bit protections from MSB. (b) MSB protection.
Figure 6.5: Reconstructed images at 1.0 b/pixel and BER = $5 \times 10^{-3}$. (a) Lena, VQ with $k' = 2 \times 2$ (26.5 dB). (b) Lena, SAPQ with $k = 2 \times 2$ and $m = 2 \times 2$ (37.2 dB). (c) Bridge, VQ with $k' = 2 \times 2$ (23.9 dB). (d) Bridge, SAPQ with $k = 2 \times 2$ and $m = 2 \times 2$ (24.7 dB).
Figure 6.6: Comparison of SAPQ and VQ for real images at 1.0 b/pixel (VQ: \( k' \times 2 \times 2 \) and \( \nu' = 16 \), SAPQ: \( k = 2 \times 2 \), \( m = 2 \times 2 \), \( n' = 8 \), and \( \eta = 4 \)).

Figure 6.7: Reconstructed Lena (VQ+Huffman with \( k' = 2 \times 2 \) and \( \nu' = 32 \) at \( R = 1.1 \) b/pixel; PSNR \( \geq 31.4 \) dB).
Figure 6.8: Reconstructed images of Lena (VQ+Huffman: $k' = 2 \times 2$ and $\nu' = 32$ at $R = 1.1$ b/pixel, SAPQ: $k = 2 \times 2$, $m = 2 \times 2$, $n' = 8$, and $\eta = 4$ at $R = 1.0$ b/pixel). (a) VQ+Huffman at BER = $10^{-4}$ (PSNR = 23.5 dB). (b) VQ+Huffman at BER = $10^{-3}$ (PSNR = 12.0 dB). (c) SAPQ at BER = $10^{-4}$ (PSNR = 31.7 dB). (d) SAPQ at BER = $10^{-3}$ (PSNR = 30.4 dB).
Huffman coder to encode the quantizer output. As can be seen in Fig. 6.7, the reconstructed image is improved over the image in Fig. 4.5(a). However, there still remains a significant contouring artifact for the VQ+Huffman scheme (as can be seen in Fig. 6.7), while no contouring artifact was observable in the SAPQ scheme (Fig. 4.5(b)). In Fig. 6.8, we compare the channel noise effect on the VQ+Huffman scheme and the SAPQ for different BERs. In the VQ+Huffman scheme of Figs. 6.8(a) and (b), we can see the error propagation and data loss problems. Due to this problem, it is even more difficult to do concealment of the corrupted images. However, for the SAPQ case, we can use the simple methods, such as the median filter or the out-range pixel smoothing schemes [51], [29], to remove the salt-and-pepper noise observable in Figs. 6.8(c) and (d).

6.5 Conclusion

In this section, Regarding the robustness of SAPQ, we demonstrate through numerical results that SAPQ has a better channel-limit performance than traditional quantizers, and show several examples for real image vector quantizations. By employing SAPQ in image quantization, we can achieve better performance in both quantization and robustness to the channel noise.
List of References


Appendix A  
Proof of Lemma 1

Let $d$ be the diameter of the convex polytope $H_1$ defined as $d := \sup \{||x - y|| : x, y \in H_1\}$. Note that $d < \infty$. Consider a sequence of cubes

$$U_t := \times_{i=1}^m \left[ -\frac{\ell}{2} - dt, \frac{\ell}{2} + dt \right], \text{ for } t = 0, 1, 2, \ldots \tag{A1}$$

and the Lebesgue measure $\mu$. $\mu(H_1) =: u$, where $u$ is a non-zero constant and $\mu(U_t) = (1 + 2dt)^m$. Suppose that the lattice $L_m$ satisfies $H_1 \supseteq U_0$.

Now we observe the number of $H_1$'s that have non-zero measured intersections with the cube $U_t$. Let $\gamma_t := \text{number of } H_i, i \in \mathbb{Z}$, such that $\mu(H_i \cap U_t) \neq 0$ and $\gamma_t^I := \text{number of } H_i, i \in \mathbb{Z}$, such that $\mu(H_i \cap U_t) = u$. Note that $\gamma_t \geq \gamma_t^I$. Then it is clear that, for $t \in \mathbb{N}$,

$$\mu(U_{t-1}) \leq \gamma_t^I u \leq \mu(U_t),$$
$$\mu(U_t) \leq \gamma_t u \leq \mu(U_{t+1}). \tag{A2}$$

Let $t_\zeta$ be the largest integer such that $\gamma_{t_\zeta} \leq \zeta$, then $t_\zeta \leq \zeta < \gamma_{t_\zeta+1}$ and $t_\zeta \to \infty$ as $\zeta \to \infty$. Thus, we have a relation

$$\frac{\gamma_{t_\zeta}}{\zeta} > \frac{\gamma_{t_\zeta}}{\gamma_{t_\zeta+1}} \geq \frac{\mu(U_{t_\zeta-1})}{\mu(U_{t_\zeta+2})} = \left[ \frac{1 + 2d(t_\zeta - 1)}{1 + 2d(t_\zeta + 2)} \right]^m \to 1, \text{ as } \zeta \to \infty. \tag{A3}$$

Since $\gamma_{t_\zeta} / \zeta \leq \gamma_{t_\zeta} / \zeta \leq 1$, for $\zeta \in \mathbb{N}$, we obtain

$$\lim_{\zeta \to \infty} \frac{\gamma_{t_\zeta}}{\zeta} = \lim_{\zeta \to \infty} \frac{\gamma_{t_\zeta}}{\zeta} = 1. \tag{A4}$$

Let $n_\zeta := \lfloor \alpha \zeta^{1/m} \rfloor$, where $[c], c \in \mathbb{R}$, is the largest integer less than or equal to $c$. First, divide the cube $U = (-a/2, a/2)^m$ into $n_\zeta^m$ cubes and denote each cube by $U_{\ell \zeta}$, for $\ell = 1, \ldots, n_\zeta^m$. Consider a sequence of equivalent lattices $L_m, t_\zeta$, where the $m$ generator vectors are obtained by multiplying $1/n_\zeta(1 + 2dt_\zeta)$ to the generator vectors of $L_m$. Then, by appropriate shifting to each cube $U_{\ell \zeta}$, we can obtain the same result of (A4). Let $H_{i, t_\zeta \ell}$ denote the convex polytope for $y_{i, t_\zeta \ell} \in L_{m, t_\zeta \ell}$ in the same manner as $H_i$ is a polytope for $y_{i} \in L_m$, where $L_{m, t_\zeta \ell}$ is a coset of $L_m$. Note that, since

$$\mu(U) \leq n_\zeta^m \zeta \mu(H_{1, t_\zeta \ell}) = \frac{\zeta}{n_\zeta} \left[ n_\zeta^m \gamma_{t_\zeta} \mu(H_{1, t_\zeta \ell}) \right] \leq \frac{\zeta}{n_\zeta} \mu(U), \text{ for } t_\zeta = 2, 3, \ldots \tag{A5}$$

and from (A4),

$$\lim_{\zeta \to \infty} n_\zeta^m \zeta \mu(H_{1, t_\zeta \ell}) = \mu(U). \tag{A6}$$

Consider codebooks $C_{i, j}$, such that $[C_{i, j}] = n_\zeta$, for $i = 1, \ldots, m$ and $j = 1, \ldots, \zeta$ in (2.7) and let $C = \bigcup_{j=1}^n (C_{1, j} \times \cdots \times C_{m, j})$. Hence we can obtain

$$\left( n_\zeta^m \right)^{2/m} \inf_{C} \left\{ \min \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i, j}} \left( X_i - y \right)^2 \right\}$$
\[ \leq (n_\gamma^n)_{2/m} \frac{1}{m} \sum_{l=1}^{n_\gamma} \sum_{j=1}^{n_\gamma} \int_{H_{l,t,t}} T \frac{||x - y||^2}{\mu(U)} dx \]
\[ \leq G(C_m)(n_\gamma^n)_{2/m} \gamma \left[ \mu(H_{l,t,t}) \right]^{1/2} / \mu(U) \]
\[ \leq G(C_m) \left[ n_\gamma^n \gamma \mu(H_{l,t,t}) \right]^{1/2} / \mu(U). \quad (A7) \]

from \( \gamma_t \leq \zeta \), where \( x = (x_1, \ldots, x_m) \in \mathbb{R}^m \), \( \int \) denotes a m-fold integral, and \( dx \) denotes \( dx_1 \ldots dx_m \).

Hence from (A6), a subsequence of \( \eta \) satisfies
\[
\limsup_{\eta \to \infty} (n_\gamma^n \gamma)^{2/m} \inf_{C} \left\{ \min_j \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i,j}} (X_i - y)^2 \right\} \leq G(C_m) \left[ \mu(U) \right]^{2/m}. \quad (A8)
\]

This completes the proof.

**APPENDIX B**

**Proof of Theorem 1**

Let \( B \) be a cube that contains the support of \( f \) and is defined by \( B := ([a,b])^m \), where \( a \) and \( b \) are finite. Consider a partition of \( B \) into \( 2^m \) cubes \( B_j \) such that \( \mu(B_j) = \frac{(b - a)/2^j}{m \in \mathbb{Z}, \ell = 1, \ldots, 2^m} \).

Define a simple function \( g_j \) as
\[
g_j(x) := \sum_{\ell=1}^{2^m} p_{\ell} I_{B_{\ell}}(x),
\]
where \( p_\ell := \sup_{x \in B_\ell} g(x) \). Then since the sequence \( (g_j(x))_j \) is monotonic and \( \lim_{\eta \to \infty} g_j(x) = g(x) \) a.e.,

it follows that \( \int g_j(x)dx \to 1 \) \( [33, p.112] \). From \( [33, p.96] \), the quantizer distortion of the SAPQ in (2.7) satisfies the relation:

\[
E \left\{ \min_j \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i,j}} (X_i - y)^2 \right\} = \int \min_j \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i,j}} (x_i - y)^2 g(x)dx
\]
\[
\leq \int \min_j \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i,j}} (x_i - y)^2 g(x)dx. \quad (B1)
\]

Let \( n_\zeta := \lfloor \alpha \zeta^{-1/m} \rfloor \) and
\[
\zeta_\ell := \left[ \zeta \cdot \frac{(p_\ell)^{\rho}}{\sum_{j=1}^{2^m} (p_j)^{\rho}} \right]. \quad (B2)
\]

Note that \( n_\zeta \zeta^{-1/m} \to \alpha \) and \( \zeta_\ell \to (p_\ell)^{\rho} / \sum_{j=1}^{2^m} (p_j)^{\rho} \) as \( \zeta \to \infty \). Using \( n_\zeta \) and \( \zeta_\ell \) make a codebook \( C_{\ell,j} \) in the same manner in Appendix A. Then, we can expand (B1) in the following way.

\[
\inf_{C} E \left\{ \min_j \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i,j}} (X_i - y)^2 \right\}
\leq \inf_{C} \int \min_j \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i,j}} (x_i - y)^2 g(x)dx
\leq \sum_{\ell=1}^{2^m} \left[ \inf_{C_\ell} \int_{B_\ell} \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{\ell,j}} (x_i - y)^2 dx \right] p_\ell \nu. \quad (B3)
\]
where \( C^t = \bigcup_{j=1}^{2^n} (C_{1,j}^t \times \ldots \times C_{m,j}^t) \). From Lemma 1 we obtain,

\[
\limsup_{\zeta \to -\infty} (n^m_\zeta \zeta_t)^{2/m} \inf \int_{B_t} \min \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i,j}^t} (x_i - y)^2 \frac{1}{v} \, dx, = G(L_m) v^{2/m},
\]

it follows that

\[
\limsup_{\zeta \to -\infty} (n^m_\zeta \zeta_t)^{2/m} \sum_{\ell=1}^{2^{m_2}} \left[ \inf_{C} \int_{B_t} \min \frac{1}{m} \sum_{i=1}^{m} \min_{y \in C_{i,j}^t} (x_i - y)^2 \frac{1}{v} \, dx \right] p_{E}\]

\[
\leq \limsup_{\zeta \to -\infty} (n^m_\zeta \zeta_t)^{2/m} \sum_{\ell=1}^{2^{m_2}} \left( \frac{1}{n^m_\zeta \zeta_t} \right)^{2/m} G(L_m) v^{2/m} \cdot p_{E} \]

\[
= \limsup_{\zeta \to -\infty} G(L_m) \sum_{\ell=1}^{2^{m_2}} \left( \frac{\zeta}{\zeta_t} \right)^{2/m} \]

\[
= G(L_m) \| g \|_p.
\]

Hence, we obtain

\[
\limsup_{q \to -\infty} \limsup_{\zeta \to -\infty} (n^m_\zeta \zeta_t)^{2/m} \inf_{C} D_{SAPQ} \leq \lim_{q \to -\infty} G(L_m) \| g \|_p = G(L_m) \| g \|_p
\]

This completes the proof.

**APPENDIX C**

**PROOF OF COROLLARY 1**

Consider an increasing sequence of cubes \( B^1 \subseteq B^2 \subseteq \ldots \subseteq B^s \). For a constant \( 0 < \lambda < 1 \), assign \( (1 - \lambda)n^m_\zeta \zeta_t^2 \) points to the cube \( B^s \) and \( \lambda n^m_\zeta \zeta_t^2 \) points to \( B^s \), which is the complement of \( B^s \). Then from [8, Theorem 2], there exists a sequence of codebook \( C_n (\in C_n \subseteq 2^n) \) such that

\[
\lim_{\lambda \to 0} \lim_{n \to \infty} (n^m_\zeta \zeta_t^2)^{2/m} \int_{B_t} \min_{y \in C_n} \| x - y \|^2 F(x) = 0.
\]

Hence, by letting \( \lambda \to 0 \), we obtain the corollary.