Project #: E-21-H17  Cost share #: E-21-343  Rev #: 4
Center #: 10/24-6-R7472-0A0  Center shr #: 10/22-1-F7472-0A0  OCA file #: 0CA
Contract#: MIP-9116113  Mod #: OPAS  Work type : RES  Document : GRANT
Prime #:  Contract entity: GTRC
Subprojects ? : N  
Main project #:  
Project unit:  ECE  Unit code: 02.010.118
Project director(s):  SMITH M J T  ECE  (404)528-7711
Sponsor/division names: NATL SCIENCE FOUNDATION / GENERAL
Sponsor/division codes: 107 / 000
Award period:  920415 to 960930 (performance) 961231 (reports)
Sponsor amount  New this change  Total to date
Contract value  0.00  169,838.00
Funded  0.00  169,838.00
Cost sharing amount  10,162.00
Does subcontracting plan apply ?: N
Title: DYNAMIC MULTIRAGE SYSTEMS FOR IMAGE & VIDEO COMPRESSION

PROJECT ADMINISTRATION DATA

OCA contact: Jacquelyn L. Bendall 894-4820
Sponsor technical contact  Sponsor issuing office
JOHN COZZENS  MARGARET A. WELLER
(703)306-1936  (703)306-1213
NATIONAL SCIENCE FOUNDATION  NATIONAL SCIENCE FOUNDATION
4201 WILSON BLVD.  4201 WILSON BLVD.
ARLINGTON, VA 22230  ARLINGTON, VA 22230

Security class (U,C,S,TS) : U  ONR resident rep. is ACO (Y/N): N
Defense priority rating : N/A  NSF supplemental sheet
Equipment title vests with: Sponsor  GIT X

Administrative comments -
ISSUED TO EXTEND TERMINATION DATE TO SEPTEMBER 30, 1996 WITH THE NSF FINAL REPORT DUE DECEMBER 31, 1996.
Closeout Notice Date: 12-SEP-1997

<table>
<thead>
<tr>
<th>Project Number</th>
<th>E-21-H17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center Number</td>
<td>10/24-6-R7472-0A0</td>
</tr>
<tr>
<td>Project Director</td>
<td>SMITH, MARK</td>
</tr>
<tr>
<td>Project Unit</td>
<td>ECE</td>
</tr>
<tr>
<td>Sponsor</td>
<td>NATL SCIENCE FOUNDATION/GENERAL</td>
</tr>
<tr>
<td>Division Id</td>
<td>3393</td>
</tr>
<tr>
<td>Contract Number</td>
<td>MIP-9116113</td>
</tr>
<tr>
<td>Prime Contract Number</td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td>DYNAMIC MULTIRANGE SYSTEMS FOR IMAGE &amp; VIDEO COMPRESSION</td>
</tr>
<tr>
<td>Effective Completion Date</td>
<td>30-SEP-1996 (Performance) 31-DEC-1996 (Reports)</td>
</tr>
</tbody>
</table>

**Closeout Action:**

<table>
<thead>
<tr>
<th>Item</th>
<th>Y/N</th>
<th>Date Submitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final Invoice or Copy of Final Invoice</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Final Report of Inventions and/or Subcontracts</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Government Property Inventory and Related Certificate</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Classified Material Certificate</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Release and Assignment</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

**Comments**

**Distribution Required:**

- Project Director/Principal Investigator: Y
- Research Administrative Network: Y
- Accounting: Y
- Research Security Department: N
- Reports Coordinator: Y
- Research Property Team: Y
- Supply Services Department: Y
- Georgia Tech Research Corporation: Y
- Project File: Y
May 11, 1993

Dr. John Cozzens
National Science Foundation
1800 G Street
Washington, DC 20550

Re: NSF Grant MIP-9116113

Dear John:

Enclosed is a copy of my annual report for my grant entitled "Dynamic Multirate Systems for Image and Video Compression." This copy, which I am sending directly to you, contains original prints (as opposed to xerox copies). Georgia Tech will probably send you a copy of my report as well but the prints will be xeroxed.

We are making good progress on this project. All of us are excited about the results we are getting. I will keep you posted as things develop.

Best regards,

Mark J. T. Smith
Associate Professor
The research proposed under this grant focuses on new directions in the areas of multirate decomposition (subband coding) and vector quantization with particular application to image and video compression. Several important developments have occurred during the first year of this project.

**FIR Spatially Adaptive Filter Banks:** Filter banks have traditionally been composed of a fixed set of contiguous bandpass filters with downsamplers and upsamplers. We have shown that it is possible to design FIR filter banks that vary in time or space and simultaneously preserve the exact reconstruction property [1]. We have also demonstrated that these FIR filters can be used in a subband image coder to improve the performance [2], [3]. Parts of this work were presented at the 1992 IEEE DSP Workshop [2], and ICASSP 1993 [3]. The ICASSP '93 paper [3] describing this work is included in the appendix as part of this report.

**IIR Spatially Adaptive Filter Banks:** This component of the research investigated computationally efficient, exact reconstruction IIR filter banks in which the filters can be varied in response to the input signal. The implementation equations for analysis and synthesis were derived along with a proof of the exact reconstruction property [4]. These filter banks can achieve a ten-fold improvement over FIR filters in computational complexity for a given magnitude response characteristic. A detailed description of this spatially adaptive IIR filter bank is given in [4], a copy of which is appended as part of this report.

It was also shown that this adaptive IIR filter bank has promising potential for subband image coding. It has the necessary degree of freedom to simultaneously preserve good magnitude response characteristics and step response characteristics. Thus it is shown in [5] that ringing distortion can be reduced in low-bit-rate subband image coding by employing these filter banks. References [4] and [5] on this subject are included in the appendix as part of this report.

**High Dimension Vector Quantization:** Crucial to any coding algorithm is the quantization process. Vector quantization has long been recognized as an attractive method of quantization. Information theory states that performance improves as the vector size is increased. Therefore, the larger the vector size, the closer one can come to truly optimal performance. However, with increasing vector size also comes exponentially increasing complexity and codebook memory. For this reason, vector quantizers have typically been limited in size to 4 × 4.

We have recently shown that jointly optimized residual vector quantization (RVQ) with multi-path searching provides an effective way to utilize large vector sizes for
image coding while at the same time maintaining manageable complexity and memory. To illustrate this, the figure on the next page shows four images of Lena, all coded at 0.25 bits/pixel using vector quantization with various vector dimensions. The first one in the upper left corner is coded with $4 \times 4$ vectors. It uses a single codebook with 16 vectors and exhaustive searching. Note that the quality is very poor at this rate; contouring and blocking effects are very visible. The second image in the upper right corner is coded with $8 \times 8$ vectors using a 4-stage RVQ with sequential searching (which is very fast). The quality is still poor but noticeably improved. Blocking artifacts from the $8 \times 8$ blocks are still visible but less pronounced. The third image in the bottom left corner uses $16 \times 16$ vectors with 16 stages and sequential searching. The quality for this case is the best of all. The blocking effects are completely gone. Detail is preserved very well and edges are very sharp.

We note here that the quality can be further improved by using multi-path searching. However, the point is that vector size provides a useful degree of freedom that has not been used very much previous due to computation and memory constraints. It is now feasible to exploit this additional latitude using RVQ because the memory and complexity issues can now be handled effectively. Several papers have been presented on the computational and memory issues, the design procedure, and various extensions of RVQ [6], [7] for image coding. A journal paper [8] describing much of this work appears in the appendix and is included as part of this report.

**Entropy-Constrained RVQ:** Residual vector quantization has the nice property of reducing both computation and memory requirements, but has the undesirable effect of degradation in performance relative to that of unconstrained single stage vector quantization (VQ). In part of the research, we show that this drawback can be overcome by exploiting the source entropy explicitly. We derive necessary conditions for optimality of this entropy-based RVQ and a new entropy-constrained RVQ (EC-RVQ) algorithm is introduced. Not only can the new EC-RVQ simultaneously outperform entropy-constrained VQ (EC-VQ) in performance, memory, and computation, but it can also be used to design high rate codebooks and/or codebooks with relatively large vector sizes (e.g. $8 \times 8$). Experimental results indicate that when the new EC-RVQ design algorithm is used, and applied to image coding, the resulting EC-RVQ achieves very high quality at low bit rates. This work was presented at ICASSP 93 [9] and is described in detail in the journal paper [10], a copy of which is included in the appendix as part of this report.

**Remarks Regarding Impact of Work**

The filter bank work is still in its early stages. However, it is clear that time-varying filter banks provide a mechanism (and the only one known to the knowledge of the PI) for handling the low bit rate distortions (ringing and aliasing) that have plagued subband image coders to date. At this point, we can demonstrate dramatic improvement for simple 1-D signals. Although some modest improvement has been shown for images (the 2-D case), the full potential of this approach has not been realized. The problem area is the adaptation control for the filter bank. We are presently concentrating our efforts on making this subsystem more robust and more accurate.
The work on residual vector quantization is further along in its development stages, although many interesting aspects of it remain to be explored. There are many new possibilities now that high dimensional vectors can be managed effectively. Most significant is that the new entropy-constrained RVQ has been shown to be capable of achieving the same quality as unconstrained entropy-constrained VQ but with a fraction of the computation and memory. We believe this will have a strong impact on coding systems because it will allow larger vector sizes to be employed. Moreover, the new EC-RVQ has extremely fast decoding capability—faster than that of fractal based decoding methods, JPEG, or virtually any other high quality method.

In the next year of research, we will build upon this base and expand these methods to video coding and very low rate image coding.

References


Appendix
EXACT RECONSTRUCTION ANALYSIS/SYNTHESIS FILTER BANKS WITH TIME-VARYING FILTERS

J. L. Arrowood Jr. and M. J. T. Smith

School of Electrical Engineering
Georgia Institute of Technology
Atlanta, GA 30332

ABSTRACT

This paper examines some of the analysis/synthesis issues associated with FIR time-varying filter banks where the filter bank coefficients are allowed to change in response to the input signal. Several issues are identified as being important in order to realize performance gains from time-varying filter banks in image coding applications. These issues relate to the behavior of the filters as transition from one set of filter banks to another occurs.

Lattice structure formulations for the time-varying filter bank problem are introduced and discussed in terms of their properties and transition characteristics.

1. INTRODUCTION

Subband coding is of great interest within the signal processing research community and is being studied from many different perspectives. Several important components are involved and are receiving attention. The two most notable are the analysis/synthesis system, which performs the frequency band splitting, and the encoding section, where the bands are quantized and coded. Most of the analysis/synthesis work has been focused on filter banks where the analysis and synthesis filters are fixed for the duration of the system’s operation. In ICASSP'92, time-varying filter banks were introduced in which the analysis and synthesis filters were allowed to change as a function of time [1].

Implementing these time-varying systems is not the trivial problem of just switching between two sets of filters on both the analysis and synthesis sides of the filter bank structure. The transition between two (or more) sets of perfect reconstruction filter banks provides a challenging reconstruction problem, since in general distortion is introduced by this process. The solution to this time-varying reconstruction problem presented in [1] involves designing multiple sets of synthesis filters that are applied while the analysis filter bank alternates between two sets of filters on the analysis side of the structure [1]. It was demonstrated that exact reconstruction could be achieved in this situation if the filters were designed properly. This was accomplished by using time domain reconstruction equations to design a set of transition synthesis filters that compensate for distortion during the switch from one set of analysis filters to another.

In [2], the application of FIR time-varying filter banks to coding images was considered. The problem addressed was the well-known inability of conventional filter banks to accurately preserve edge characteristics at low coding rates. In particular, when images are coded at low bit rates, ringing distortion at object edges is often observed. This is due to the step response characteristics of the analysis/synthesis filters and is a consequence of their having good magnitude response properties. If filters with monotonic step response characteristics are used (which precludes their having good magnitude response characteristics) aliasing distortion becomes very visible. Time-varying filter banks have the ability to simultaneously reduce the aliasing distortion and ringing distortion at low bit rates. As discussed in [2], this can be done by switching back and forth between analysis filters with different spectral and temporal (step response) characteristics such that in regions where no major transitions occur, the filter set with good magnitude response characteristics is used. When transition regions or object edges are encountered, the system switches to the filter set with good step response properties.

The time-domain formulation of the problem [1] is such that each switch in analysis filters requires that many synthesis filters be applied sequentially. If we consider the case with 16-tap filters and assume that we are free to switch back and forth between analysis filters without restriction, 256 synthesis filters are necessary to exactly reconstruct the input, all of which are interdependent. To reduce the severity of this problem, constraints were imposed on the frequency in which the filters could be alternated. Even with this, 46 synthesis filters were needed to guarantee exact reconstruction. The large number of synthesis filters can make it difficult to design the system since all filters are designed together.

The lattice formulation which is introduced in the next section utilizes a structure that guarantees exact reconstruction in the presence of changing filter coefficients. This approach has the advantage of avoiding the problem of having to maintain large sets of synthesis reconstruction filters. However, these transition areas can be troublesome for coding applications, and care must be taken as to what method is used when switching between sets of coefficients as discussed in the subsequent sections.

This work is supported in part by the National Science Foundation under contract MIP-9116113 and the National Aeronautics and Space Administration.
FILTER BANKS

This structure commonly presented in literature is that the general case

Figure 1: Efficient lattice implementation.

2. TIME-VARYING, LATTICE-BASED FIR FILTER BANKS

The lattice structure for two channel conventional FIR filter banks can achieve exact reconstruction in the absence of coding [3]. This structure can be applied to time-varying filter banks as well. In the following derivation, we consider the general case as well as some more specific cases for example. Figure 1 shows the general structure needed to set up the lattice based FIR filter bank. For each of the lattice stages the following holds:

\[ R_i = \begin{bmatrix} \alpha_{i0} & \alpha_{i2} \\ \alpha_{i1} & \alpha_{i3} \end{bmatrix} \]

\[ E_i = R_i^{-1} \]

Figure 2 shows how this matrix is implemented to form the lattice. The difference in the lattice shown here and the one commonly presented in literature is that the actual inverse of \( R_i \) is used for \( E_i \), instead of the transpose. This maintains the perfect reconstruction property when the lattices are being exchanged to form the time-varying structure by ensuring that the following always holds:

\[ R_i \times E_i = I \]

where \( I \) is the identity matrix. The overall structure can be made time-varying at this juncture simply by allowing any or all of the matrices, \( R_i \), to change as a function of time. As long as the corresponding \( E_i \) also changes at the appropriate time in the synthesis section, the overall structure remains perfectly reconstructing.

It is important that the switch in the synthesis section take place at the correct time given a change in the analysis portion of the structure. First, the implementation in Figure 1 specifies the number of lattice stages at \( J+1 \), leading to filters of length \( N = 2 \times (J+1) \) being produced. Furthermore, this implementation requires that if the matrix \( R_i \) is switched out with a new matrix, \( r_i \), at time \( n_0 \), one needs to change the lattice matrix \( E_i \) to \( e_i \) (the inverse of \( r_i \)) at the time \( n_0 + (J - i) \). The delay is necessary due to the delays found between lattice matrices. Changing out synthesis lattices with respect to analysis lattices in this fashion will maintain the PR (perfect reconstruction) property of the system, while allowing the filters “built” by the subsequent implementation of the lattices to change.

For the case where the matrices are unitary, and real coefficients are used, a more efficient implementation can be used. Here the matrices \( R_i \) are given by:

\[ R_i = \begin{bmatrix} 1 & \alpha_i \\ -\alpha_i & 1 \end{bmatrix} \]

The paraunitary nature of the filter bank forces the impulse response coefficients of the filter bank to be related as:

\[ h_1[n] = (-1)^n h_0[(N-1) - n] \]
\[ g_0[n] = h_0[(N-1) - n] \]
\[ g_1[n] = h_1[(N-1) - n] \]

where \( N \) is the length of the filters created and \( g_0 \) and \( g_1 \) are the corresponding synthesis filters. It should be mentioned here that the above equations hold only for regions where the filters are stationary—transition filters will not necessarily maintain the above properties. Note also that the length of each of the filters is \( 2(J+1) \) and that all four filters are expressible uniquely in terms of \( \alpha_0, \ldots, \alpha_J \).

From the discussion above, it is easy to gather that a different set of four filters can be implemented simply by specifying a new set of \( J+1 \) lattice coefficients. If these new coefficients are specified as \( \beta_0, \ldots, \beta_J \), one needs only to develop a reasonable method for changing from the \( \alpha \)'s to the \( \beta \)'s and vice versa in order to implement a time-varying filter bank that can switch between two sets of analysis/synthesis filters.

The transition between the two sets must be done with care, since it is important that the impulse response of the transition filters provide a relatively smooth change in going from one set to the other. The type of method used should depend on the type of filters being interchanged. Obviously, filter sets that are similar in nature, and thus have similar coefficients, are easier to switch between than filters that are markedly different. One method for switching between the
two that provides decent results comes from the paraunitary property of the lattice itself. Assume, for example, that the transition is being made between the filter set based on the \( \alpha \)'s to the one based on the \( \beta \)'s. The first step, at time \( n_0 \), would be to “turn off” \( \alpha \) through \( \alpha_1 \), replacing each coefficient with a zero, while at the same time replacing \( \alpha_0 \) with \( \beta_0 \). The remaining \( \beta \)'s are “turned on” in a cascade fashion, with \( \beta_1 \) being turned on at time \( n_1 \), then \( \beta_2 \) at time \( n_2 \) and so on, until all of the lattice stages are functioning with the new coefficients. This simple procedure will keep the transitions somewhat smooth, for many cases. This same procedure will work in the opposite direction, when the coefficients are switched from the \( \beta \)'s to the \( \alpha \)'s. Figure 3 shows some of the transition filters produced when the system goes from a broad to a sharp filter using the cascade method described above. For Figure 4, a change is made between the same two sets of coefficient, except that all of the coefficients were swapped out simultaneously, with the result being the poor transition filters that are seen. Thus, care should be taken when implementing the time-varying structure using the lattice approach, since the transition filters may not automatically turn out to be desirable.

A comparison can also be made with respect to the time-domain formulation versus the lattice structure in terms of the transition filter quality. Figure 5 displays several of the transition filters that must be used, and therefore stored, in the synthesis side of the filter bank implemented using the methods discussed in [1]. Figure 6 shows what happens in the same region for the lattice structure, with the difference being that the lattice structure only requires the methodical swapping out of two sets of coefficients.

3. APPLICATION TO SUBBAND IMAGE CODING

The time-varying FIR analysis/synthesis system can be used in a subband image coder to reduce the effects of aliasing
and ringing distortion. The approach considered here involves a separable implementation of the analysis/synthesis system in which the rows are split first with the filter bank and then the columns of the result are split to form four subband images. The band splitting can be continued until the desired number of subband images is obtained. As with conventional subband coders, the bits are allocated according to the importance of each individual subband. For natural images, the low frequency subband images are most important and the high frequency ones are less important. Thus at low rates, information in the high frequency bands is often lost completely.

As an example of the possibilities of the time-varying structure, consider the image example found in Figure 7. For this figure the top portion was used as input to both a standard, non-time-varying system and to a time-varying system based on the time-domain formulation. The subband outputs were quantized. The low frequency channel was represented with 5 bit uniform quantization. The high frequency channel represented with only 3 bits. Clearly, the ringing, evidenced by the varying bands of shading, is reduced in the time-varying case with respect to the output of the standard filter bank.

4. CLOSING COMMENTS

The idea of time-varying filter banks is still very new and much remains to be understood before the full potential of this approach is realized. An important aspect of the problem which has not been studied sufficiently at this time is the design of an adaptation strategy that reflects both the analysis and transition synthesis filter characteristics. In homogeneous regions of the input image, filter banks with good magnitude response characteristics are known to work well and should be used. When an object edge is encountered, analysis and synthesis filters with good step response characteristics are desirable to avoid ringing distortion. The problem is that after a change in analysis filters, a sequence of synthesis filters follows. Unlike the analysis filter sets which are changed directly to match step and magnitude responses to the input, this flexibility is not present in selecting the synthesis filter sets. It is expected that better results are possible with an adaptation strategy and design procedure that allow both analysis and synthesis filter characteristics to be optimized to the input signal. These issues are currently being examined. Some examples of coded images will be presented at the conference that reflect our latest efforts in this direction.

5. REFERENCES

Recursive Time-Varying Filter Banks for Subband Image Coding

Mark J. T. Smith and Wilson C. Chung  
Georgia Institute of Technology  
School of Electrical Engineering  
Atlanta, Georgia 30332

Abstract

Filter banks and wavelet decompositions that employ recursive filters have been considered previously and are recognized for their efficiency in partitioning the frequency spectrum. This paper presents an analysis of a new infinite impulse response (IIR) filter bank in which these computationally efficient filters may be changed adaptively in response to the input. The filter bank is presented and discussed in the context of finite-support signals with the intended application in subband image coding.

In the absence of quantization errors, exact reconstruction can be achieved and by the proper choice of an adaptation scheme, it is shown that IIR time-varying filter banks can yield improvement over conventional ones.

1 Introduction

Subband image coding is a well-known technique in which an input is split into a small number of subband images, each of which is associated with a different region in the spatial frequency plane. The subband images are decimated to their Nyquist rates prior to being quantized and coded. In reconstruction the images are decoded, upsampled, and then merged together. The process in which the input is split and merged is called analysis/synthesis. There are many performance and systems related design issues associated with analysis/synthesis as discussed in [2], [4], [7] and elsewhere. Recursive analysis and synthesis filter banks have been shown to be particularly attractive because they can be designed to reconstruct the input exactly in the absence of coding and can achieve tremendous computational efficiency relative to comparable FIR systems [4], [5]. These filter banks are typically two-band systems that serve as building blocks for uniform and non-uniform band tree structured systems. However, parallel form recursive filter banks have also been shown to work well [11]. In such a case, the subband images are complex valued. Thus to avoid increasing the data rate, Hermitian symmetry is exploited to counter the sample increase due to the presence of complex numbers in the subband images.

When subbands are quantized, distortion results and the degree of the distortion is inversely proportional to the bit rate. The quantization noise generates three types of interrelated distortions:

1. Aliasing. This is due to the partial loss of aliasing cancellation in the synthesis section that arises from quantization. It appears subjectively as blurring in the reconstructed images.

2. Ringing distortion. This appears as amplitude oscillations (or ringing) in the vicinity of edges in the image. As discussed in [4], its cause may be directly related to the ripples in the step response characteristics of the analysis/synthesis filters.

3. In-band spectral distortion. This distortion is attributed to the deviation in the spectral magnitude and phase of subband images excluding the inter-band effects of aliasing and ringing. In essence, this form of distortion accounts for all remaining deviations when aliasing and ringing effects are removed.

---

1 This work was supported in part by the National Science Foundation under contract MIP-9116113 and the National Aeronautics and Space Administration.
Much attention has been given to minimizing the perceptually important coding distortions in image coding systems. For low bit rate subband image coding systems, aliasing and ringing distortions are often cited as being objectionable. Efforts to reduce these effects have been considered previously but with limited success [4], [5], [12]. The difficulty is that the ripples of the step response of the filters, which are the source of the ringing distortion, are needed to achieve good filter magnitude response characteristics. Good spectral magnitude characteristics are important for reducing the effects of aliasing in the presence of coding. Aliasing distortion appears as blurring in the image and is subjectively noticeable. If filters with good magnitude characteristics are used to reduce the aliasing, then ringing distortion results that is noticeable and objectionable. As to which is worse is a matter of opinion. It is clear, however, that the elimination or reduction of both is desirable. This dilemma is a consequence of the Gibbs Phenomenon which implies that it is impossible to have filter banks with good (i.e. monotonic) step response characteristics and good magnitude response characteristics within the conventional paradigm. However, it is possible to overcome this seemingly fundamental difficulty by using time varying filter banks as we shall see. The first introduction of the concept of time-varying filter banks appears in [3] and is based on the time domain theory and design methodology presented in [15], [14], and [13]. The approach taken in [3] has some attractive features: time-varying capability is conceptually simple in this framework and system delay can be controlled very easily. On the other side of the coin, the approach is inherently limited to FIR systems and the system design task can be difficult for practical filters.

In this paper, we introduce the concept of time-varying recursive filter banks and demonstrate that they provide the additional degree of freedom that allows them to overcome the step-response-magnitude-response constraint associated with the Gibbs Phenomenon. We derive the conditions under which we can switch between filters of desirable properties. Furthermore, this new time-varying recursive filter preserves the exact reconstruction property and maintains very high computational efficiency. We presented this idea at the 1992 DSP Workshop [16] and discussed some of the analysis issues at the 1992 Asilomar Conference [17]; both [16] and [17] are rather limited due to page length constraints.\footnote{The authors recently became aware of similar work developed independently in Norway by Husoy and Aase [18].}

This paper begins with a discussion of the exact reconstruction IIR filter bank after which the time-varying conditions are derived. The equations for time-varying analysis/synthesis are derived and a design strategy is developed that allows for much improved performance over conventional filter banks.

## 2 Exact Reconstruction IIR Filter Banks

The starting point for this work is the two-band analysis/synthesis system shown in Figure 1 where \( \{H_0(z), H_1(z)\} \) and \( \{G_0(z), G_1(z)\} \) are the pairs of lowpass/highpass analysis/synthesis...
The analysis filters have the form:

\[
H_0(z) = P_0(z^2) + z^{-1}P_1(z^2) \\
H_1(z) = P_0(z^2) - z^{-1}P_1(z^2)
\]  

and may be implemented using the well-known polyphase structure shown in Figure 2 where \(P_0(z)\) and \(P_1(z)\) are the analysis polyphase filters and \(Q_0(z)\) and \(Q_1(z)\) are the synthesis polyphase filters.

The exact reconstruction property of this recursive filter bank is evident from an examination of the polyphase structure. Observe that the criss-cross networks in the analysis and synthesis sections of Figure 2 are simply two-point DFT butterflies and that the cascade of the two results in an identity system. Therefore the combined analysis/synthesis polyphase system can be simplified as shown in Figure 3. By inspecting Figure 3, it is clear that if

\[
Q_0(z) = \frac{1}{P_0(z)} \quad \text{and} \quad Q_1(z) = \frac{1}{P_1(z)}
\]

then the overall system reduces to an identity system and hence the reconstruction is exact.

A few points are noteworthy in this regard. First, \(P_0(z)\) and \(P_1(z)\) are assumed to be causal.
stable recursive filters with zeros strictly outside the unit circle and have allpass or near allpass characteristics. Second, by definition, the synthesis polyphase filters $Q_0(z)$ and $Q_1(z)$ will have their poles outside the unit circle and their zeros inside the circle. Consequently, they are anti-causal and stable. Third, because the image boundaries are finite in length, the circular convolution method discussed in [4] is used to handle the analysis and reconstruction at the image boundaries in order to avoid an increase in the subband data. It should be noted that the recursive polyphase filters are well-behaved and do not attempt to invert stopbands or magnify selected regions of the spectrum that might lead to adverse emphasis of coding noise [4]. Comparisons were given to systems based on linear-phase FIR filters and to those based on linear-phase and non-linear-phase IIR filters. The interested reader is directed to this reference [4] for further details.

3 The New Time-Varying IIR Filter Banks

In this section, we derive a new variant of the IIR filter bank in which the coefficients of the constituent filters are allowed to vary with time. The coefficients of the polyphase filters $P_0(z)$ and $P_1(z)$ are selectively changed at some point in the filtering process to another set of polyphase filters. This process of changing the filters allows us to improve the coding performance. The problem we address in this section is how to reconstruct without distortion once we have switched the filter coefficients. By examining the simplified polyphase structure in Figure 3, it is apparent that it is sufficient to determine the conditions under which $r_i[n] = t_i[n]$ for $i = 1, 2$. Clearly if we can guarantee that $r_i[n] = t_i[n]$ at all times before, during, and after switching the coefficients, then exact reconstruction is also guaranteed.

To illustrate the analysis and reconstruction problem, consider the graphical illustration shown in Figure 4. Assume that the impulse response $p_L[n]$ shown in the figure corresponds to that of a causal, stable analysis polyphase filter

$$P_L(z) = \frac{\alpha_0^L + \alpha_1^L z^{-1} + \cdots + \alpha_{M_0}^L z^{-M_0}}{1 + \beta_1^L z^{-1} + \beta_2^L z^{-2} + \cdots + \beta_{N_0}^L z^{-N_0}}.$$  

At time $n = 0$, the filter is changed to $p_R[n]$ corresponding to a new polyphase filter

$$P_R(z) = \frac{\alpha_0^R + \alpha_1^R z^{-1} + \cdots + \alpha_{M_1}^R z^{-M_1}}{1 + \beta_1^R z^{-1} + \beta_2^R z^{-2} + \cdots + \beta_{N_1}^R z^{-N_1}}.$$  

The labels $L$ and $R$ signify that the signal or filter coefficient is associated with the left and right halves of the time index, respectively.

3.1 Time Domain Analysis

The direct form difference equations that implement the polyphase filters are:

$$v_L[n] = \sum_{m=0}^{M_0} \alpha_m^L x[n - m] - \sum_{\ell=1}^{N_0} \beta_{\ell}^L v_L[n - \ell]$$  

$$- \infty < n \leq -1.$$  

Figure 3: Two-Band Polyphase Implementation with DFT Butterflies Omitted
and
\[ v_R[n] = \sum_{m=0}^{M_1} \alpha_m^R x[n-m] - \sum_{t=1}^{N_1} \beta_t^R v_R[n-t] \quad 0 \leq n < \infty \]  \hspace{1cm} (4)

Here, \( v_L[n] \) and \( v_R[n] \) are the left-sided and right-sided components of the output \( v[n] \), i.e. \( v[n] = v_L[n] + v_R[n] \). In addition, equation (4) requires the initial conditions \( v_R[-N_1], v_R[-N_1 - 1], \ldots, v_R[-1] \) in order to evaluate \( v_R[0] \) since
\[ v_R[0] = \sum_{m=0}^{M_1} \alpha_m^R x[-m] - \sum_{t=1}^{N_1} \beta_t^R v_R[-t] \]  \hspace{1cm} (5)

Rather than making these initial conditions zero, which effectively discards the recent history of the input, we assume the more desirable initial conditions
\[ v_R[n] = v_L[n] \quad \text{in the range} \quad -N_1 \leq n \leq -1. \]

Generalizing the reconstruction procedure discussed in [4] for conventional recursive filter banks, we obtain the reconstruction shown in Figure 5 which is based on anti-causal filtering. Here \( q_R[n] \) and \( q_L[n] \) are impulse responses corresponding to the stable, anti-causal polyphase synthesis filters. The corresponding difference equations are
\[ x_R[n - A] = \frac{1}{-\alpha_{M_1}^R} \sum_{t=1}^{M_1} \alpha_{M_1-t}^R x_R[n+t-A] + \frac{1}{\alpha_{M_1}^R} \sum_{m=0}^{N_1} \beta_m^R v_R[n+m-B] \]  \hspace{1cm} (6)

for \( 0 \leq n < \infty \) and
\[ x_L[n - C] = \frac{1}{-\alpha_{M_0}^L} \sum_{t=1}^{M_0} \alpha_{M_0-t}^L x_L[n+t-C] + \frac{1}{\alpha_{M_0}^L} \sum_{m=0}^{N_0} \beta_m^L v_L[n+m-D] \]  \hspace{1cm} (7)

for \( -\infty < n \leq -1 \), where \( A, B, C, \) and \( D \) are arbitrary integer shift constants. The reconstruction sections, \( x_R[n] \) and \( x_L[n] \) are right-sided and left-sided components of the reconstructed signal \( x[n] \), as shown in Figure 5 and have the property that
\[ x[n] = x_R[n] + x_L[n]. \]  \hspace{1cm} (8)

Although the form of the reconstruction is given by equations (3- 7), the unknown parameters \( A, B, C, \) and \( D \) must be determined.

### 3.2 z-Transform Domain Analysis

In this section, we determine the exact reconstruction conditions under this new time-varying filter bank paradigm (i.e. find \( A, B, C, \) and \( D \)) and show explicitly that reconstruction is exact.
The time-varying filter bank problem can be analyzed in the $z$-transform domain in terms of a causal and anti-causal unilateral $z$-transform. We define the causal unilateral $z$-transform as

$$X_R(z) = \sum_{n=0}^{\infty} x[n]z^{-n}$$  \hspace{0.5cm} \text{(9)}$$

and the anti-causal unilateral $z$-transform as

$$X_L(z) = \sum_{n=-\infty}^{-1} x[n]z^{-n}.$$  \hspace{0.5cm} \text{(10)}$$

The sum of $X_R(z)$ and $X_L(z)$ is identical to the bilateral $z$-transform of $x[n]$. These causal and anti-causal transforms have shift properties associated with them that may be described in the following way. Suppose that $n_0$ is an integer and $u[n]$ is the unit step function. Then, for a time shift of $n_0$ in $x[n]$,

$$w[n] = x[n - n_0]$$

the causal unilateral $z$-transform $W_R(z)$ is

$$W_R(z) = z^{-n_0}X_R(z) + \sum_{\ell=0}^{n_0-1} x[\ell - n_0]z^{-\ell}u[n_0 - 1] - z^{-n_0} \sum_{\ell=0}^{-n_0-1} x[\ell]z^{-\ell}u[-n_0 - 1].$$  \hspace{0.5cm} \text{(11)}$$

Similarly, for the anti-causal $z$-transform, the same time shift

$$w[n] = x[n - n_0]$$

results in the anti-causal transform

$$W_L(z) = z^{-n_0}X_L(z) - z^{-n_0} \sum_{\ell=1}^{n_0} x[-\ell]z^{\ell}u[n_0 - 1] + z^{-n_0} \sum_{\ell=0}^{-n_0-1} x[\ell]z^{\ell}u[-n_0 - 1].$$  \hspace{0.5cm} \text{(12)}$$

This shift property will be used to take the transforms of the analysis equations (3) and (4) and the synthesis equations (6) and (7). A derivation of this shift property is given in appendix A.

Using the causal and anti-causal $z$-transforms and their corresponding shift properties, we can take the $z$-transforms of the analysis and synthesis equations. The transform properties allow us to model the initial conditions explicitly for the point at which we switch the filter coefficients. In this analysis, we assume that we switch the filter coefficients at $n = 0$.

For convenience in taking the transform, we express equation (4) as

$$\sum_{\ell=0}^{N_1} \beta_{RL}^{R}[n - \ell] = \sum_{m=0}^{M_1} \alpha_{RL}^{R}[n - m] \quad 0 \leq n < \infty$$  \hspace{0.5cm} \text{(13)}$$

(where $\beta_{RL}^{R} = 1$) and compute its $z$-transform using the definition (9) and shift property (11). We obtain

$$\sum_{\ell=0}^{N_1} \beta_{RL}^{R} z^{-\ell}V_R(z) + \sum_{\ell=1}^{N_1} \alpha_{RL}^{R}[k - \ell]z^{-k} = \sum_{m=0}^{M_1} \alpha_{RL}^{R}[n]z^{-m} + \sum_{m=1}^{M_1} \alpha_{RL}^{R}[n]z^{-m}$$  \hspace{0.5cm} \text{(14)}$$
Next we can write the right-sided synthesis equation (equation (6)) as

\[ \sum_{\ell=0}^{M_1} x_R[n + \ell - A] \alpha_{M_1-\ell}^{R} = \sum_{m=0}^{N_1} \beta_{N_1-m}^{R} v_R[n + m - B] \]  

and take its z-transform to obtain

\[ \sum_{\ell=0}^{M_1} \alpha_{M_1-\ell}^{R} x_R(z) z^{-A} + \sum_{\ell=0}^{M_1} \alpha_{M_1-\ell}^{R} \left( \sum_{m=0}^{A-\ell-1} x_R[m - A + \ell] z^{-m} u[A - \ell - 1] \right) \]

\[ -z^{-A+\ell} \sum_{m=0}^{A-\ell-1} x_R[m] u[-A + \ell - 1] \]

\[ = \sum_{m=0}^{N_1} \beta_{N_1-m}^{R} v_R(z) z^{-m-B} + \sum_{m=0}^{N_1} \beta_{N_1-m}^{R} \left( \sum_{\ell=0}^{B-m-1} v_R[\ell - B + m] z^{-\ell} u[B - m - 1] \right) \]

\[ - z^{B-m} \sum_{\ell=0}^{B-m-1} v_R[\ell] u[-B + m - 1]. \]  

(16)

This is the z-transform of the synthesis equation for the right-sided sequence \( x_R[n] \). If the right and left sides of this equation can be shown to be identical, then we have shown exact reconstruction for \( x_R[n] \). To explicitly show that the equality holds, we must express \( V_R(z) \) in terms of \( X_R(z) \), the initial conditions, and the filter coefficients. Using equation (14), we can remove the \( V_R(z) \) term by substituting

\[ z^{N_1-B} \left( \sum_{m=0}^{M_1} \alpha_m^{R} x_R(z) z^{-m} + \sum_{m=1}^{M_1} \alpha_m^{R} \sum_{\ell=0}^{m-1} x_R[\ell - m] z^{-\ell} - \sum_{\ell=1}^{N_1} \beta_{\ell-1}^{R} \sum_{k=0}^{\ell-1} v_R[k - \ell] z^{-k} \right) \]

for

\[ \sum_{m=0}^{N_1} \beta_{N_1-m}^{R} V_R(z) z^{-m-B} \]

in equation (16). After the substitution, equation (16) becomes

\[ \sum_{\ell=0}^{M_1} \alpha_{M_1-\ell}^{R} x_R(z) z^{-A} + \sum_{\ell=0}^{M_1} \alpha_{M_1-\ell}^{R} \left( \sum_{m=0}^{A-\ell-1} x_R[m - A + \ell] z^{-m} u[A - \ell - 1] \right) \]

\[ - \sum_{\ell=0}^{M_1} \alpha_{M_1-\ell}^{R} z^{-A+\ell} \sum_{m=0}^{A-\ell-1} x_R[m] u[-A + \ell - 1] \]

\[ = z^{N_1-B} \sum_{m=0}^{M_1} \alpha_m^{R} x_R(z) z^{-m} + z^{N_1-B} \sum_{m=1}^{M_1} \alpha_m^{R} \sum_{\ell=0}^{m-1} x_R[\ell - m] z^{-\ell} - z^{N_1-B} \sum_{\ell=1}^{N_1} \beta_{\ell-1}^{R} \sum_{k=0}^{\ell-1} v_R[k - \ell] z^{-k} \]

\[ + \sum_{m=0}^{B-m-1} v_R[\ell - B + m] z^{-\ell} u[B - m - 1] \]

\[ - z^{B-m} \sum_{\ell=0}^{B-m-1} v_R[\ell] u[-B + m - 1]. \]  

(18)
By examining this equation, it becomes apparent that we can group like terms and that there must be equivalence within these groups in order for the equality to hold. Consider, first, terms 6, 7, and 8, which are the only terms containing \( v_R[n] \) components. The sum of these terms must be zero, i.e.

\[
-z^{N_1-B} \sum_{t=1}^{N_1} \beta_t^R \sum_{k=0}^{t-1} v_R[k - \ell] z^{-k} + \sum_{m=0}^{N_1} \beta_{N_1-m}^R \sum_{t=0}^{B-m-1} v_R[\ell - B + m] z^{-t} u[B - m - 1] \\
- \sum_{m=0}^{N_1} \beta_{N_1-m}^R z^{B-m} \sum_{t=0}^{B+m-1} v_R[\ell] u[-B + m - 1] = 0
\]  

(6)  

(7)  

(8)

At initial glance, it is apparent that cancellation can only occur between either terms 6 and 7 or 7 and 8 because these are the only combinations in which the signs are opposite. A closer examination shows that terms 7 and 8 are disjoint in the variable \( m \) due to the \( u[-B + m - 1] \) terms. Thus, it is sufficient to focus on terms 6 and 7. In term 6, if we let \( \ell = N_1 - m \) we obtain

\[
-z^{N_1-B} \sum_{m=0}^{N_1-1} \beta_{N_1-m}^R \sum_{k=0}^{N_1-m-1} v_R[k - N_1 + m] z^{-k}
\]

which cancels with term 7 when \( B = N_1 \). Moreover, when \( B = N_1 \), term 8 disappears because \( u[-B + m - 1] \) goes to zero.

We can now group terms 1 and 4 together in equation (18) because only they contain the \( X_R(z) \) components. Letting \( \ell = M_1 - m \) in term 1, we obtain

\[
\sum_{m=0}^{M_1} \alpha_m^R X_R(z) z^{M_1-m-A}
\]

This becomes identical to term 4 when \( A = M_1 \), resulting in cancellation.

The only remaining terms are 2, 3, and 5. Immediately we observe that term 3 is zero. Note that because \( A = M_1 \), the \( u[-A + \ell - 1] \) forces term 3 to be zero. Thus all that is left is to show that terms 2 and 5 cancel. Letting \( \ell = M_1 - n \) in term 2, we obtain

\[
\sum_{n=0}^{M_1} \alpha_n^R \sum_{m=0}^{n-1} x_R[m - n] z^{-m} u[n - 1] = \sum_{n=1}^{M_1} \alpha_n^R \sum_{m=0}^{n-1} x_R[m - n] z^{-m}
\]

which is equivalent to term 5. Thus we see that when \( A = -M_1 \) and \( B = -N_1 \), we can exactly reconstruct \( x_R[n] \).

What remains is to prove that we can reconstruct \( x_L[n] \). The same procedure can be applied to the left-sided equations. In particular, the analysis equation (3) can be written as

\[
\sum_{\ell=0}^{N_0} \beta_{\ell}^L v_L[n - \ell] = \sum_{m=0}^{M_0} \alpha_m^L x[n - m]
\]

(20)

where \( \beta_0^L = 1 \). Its corresponding \( z \)-transform is

\[
\sum_{\ell=0}^{N_0} \beta_{\ell}^L z^{-\ell} V_L(z) - \sum_{k=1}^{N_0} \beta_{\ell}^L z^{-\ell} \sum_{\ell=0}^{\ell} v_L[-k] z^k u[\ell-1] = \sum_{m=0}^{M_0} \alpha_m^L z^{-m} X_L(z) - \sum_{m=0}^{M_0} \alpha_m^L z^{-m} \sum_{\ell=1}^{\ell} x_L[\ell] z^\ell u[m-1]
\]

(21)
The $z$-transform for the left-sided synthesis equation (7) is

$$
\sum_{t=0}^{M_0} \alpha_{M_0-t}^L X_L(z) z^{-t-C} - \sum_{k=1}^{C-t} \sum_{t=0}^{M_0} x_L[-k] z^{t} u[C - \ell - 1] + \sum_{k=0}^{t-C-1} x_L[k] z^{-k} u[\ell - C - 1]) =

\sum_{m=0}^{N_0} \beta_{N_0-m}^L z^{-m-D} V_L(z) - \sum_{m=0}^{N_0} \sum_{t=0}^{D-m} v_L[-\ell] z^{t} u[D - m - 1] + \sum_{t=0}^{D-m-1} v_L[\ell] z^{-t} u[m - D - 1])
$$

(22)

Applying equation (21) to equation (22), we can remove the $V_L(z)$ components and we obtain

$$
\sum_{t=0}^{M_0} \alpha_{M_0-t}^L X_L(z) z^{-t-C} - \sum_{k=0}^{t-C-1} x_L[k] z^{-k} u[\ell - C - 1]) =

z^{N_0-D} \sum_{t=0}^{N_0} v_L[-\ell] z^{t} u[D - m - 1] + \sum_{t=0}^{M_0} \alpha_{M_0-t}^L z^{-t} X_L(z)

- \sum_{m=0}^{M_0} \alpha_{m}^L z^{-m} x_L[-\ell] z^{t} u[m - 1] - \sum_{m=0}^{N_0} \beta_{N_0-m}^L z^{-m-D} (\sum_{t=0}^{D-m} v_L[-\ell] z^{t} u[m - D - 1])
$$

(23)

Again we can equate like terms and solve for the unknowns. Doing so, we find $C = M_0$ and $D = N_0$, and consequently prove that $x_L[n]$ may be exactly reconstructed.

4 Results

The time-varying filter bank reconstruction problem shares similarities with conventional inverse polyphase reconstruction. However, the interactions at the filter coefficient switching points make the analysis very different for the time-varying case. Analysis and interpretation of time-varying filter banks is complicated by the fact that the tools we would like to use, such as the DTFT and $z$-transform, are inherently time-invariant. By couching the problem in terms of left and right signal components and the initial conditions associated with them, we have proved that maximally decimated time-varying filter bank analysis/synthesis systems can be exactly reconstructing. We determined that the unknown parameters in the analysis equations (3) and (4) and synthesis equations (6) and (7) are

$$
A = M_1
B = N_1
C = M_0
D = N_0
$$

and that the resulting reconstruction synthesis equations are

$$
x_R[n - M_1] = \frac{1}{-\alpha_{M_1}} \sum_{t=1}^{M_1} \alpha_{M_1-t}^R x_R[n + \ell - M_1] + \frac{1}{\alpha_{M_1}} \sum_{m=0}^{N_1} \beta_{N_1-m}^R v_R[n + m - N_1]
$$

(25)
for $0 \leq n < \infty$ with $v_R[n] = v_L[n]$ in the range $-N_1 \leq n \leq -1$ and

$$x_L[n-M_0] = \frac{1}{\alpha_{M_0}^L} \sum_{\ell=1}^{M_0} \alpha_{M_0-\ell}^L x_L[n+\ell-M_0] + \frac{1}{\alpha_{M_0}^L} \sum_{m=0}^{N_0} \beta_{N_0-m}^L v_L[n+m-N_0] \quad (26)$$

for $-\infty < n \leq -1$.

### 4.1 Constraints on Switching in the System

This $z$-domain analysis and the resulting synthesis equations reveal some interesting and important properties of the analysis/synthesis system. To begin, observe that the derivation leading to equation (25) proves that the samples of the input $x[n]$ in the range $-M_1 \leq n \leq \infty$ can be reconstructed using equation (25), where $x[n] = x_R[n]$. Similarly the samples of $x[n]$ in the range $-\infty \leq n < -M_0$ can be reconstructed using equation (26), where $x[n] = x_L[n]$ in this region.

In order to reconstruct all sample values using equations (25) and (26), constraints upon the variables $M_0$ and $M_1$ must be imposed. This observation can be drawn from Figure (6) with the following constraint being placed upon $M_0$ and $M_1$:

$$M_1 + 1 \geq M_0.$$ 

If this constraint is not met, the values in between $M_0$ and $M_1$ cannot be recovered from the reconstruction equations (25) and (26) as shown in Figure (6). There exists a gap of unrecoverable samples in between $M_0$ and $M_1$.

Equations (25) and (26) also show that by supplying the original sample values in the transition regions explicitly, the constraint on the separation between switching points can be removed completely. As we saw, switching coefficients at an interval less than or equal to $M_1$ precludes the recovery of all samples in the region $-M_1 \leq n < 0$. If these missing samples were supplied externally, then reconstruction can proceed without difficulty. Unfortunately, such an approach results in a data increase, i.e. the analysis/synthesis system is no longer maximally decimated.

For convenience of being able to use the conventional unilateral $z$-transform, the analysis has been given under the assumption that the switching of filters occurs at $n = 0$. But nothing is changed by switching the coefficients at an arbitrary time, $n = n_0$, except for a time shift. All the equations derived are applicable to a system with arbitrary time shift at $n = n_0$. Based on the analysis, it should be clear that these results hold independent of when the filters are switched.

However, the equations reveal limits on how frequent these switching of the filter coefficients can occur within a given interval and on the numerator/denominator orders of the polyphase filters. To illustrate this, consider the example in Figure (6). Equation (25) reconstructs valid samples of $x_R[n] = x_L[n]$ for $n = -1, -2, \ldots, -M_1$. These samples are used as initial conditions in equation (26). However, if another switch of coefficients occurs at time $n = K$ where $-M_1 < K < 0$, only $K$ valid samples can be reconstructed from equation (25), the remaining samples in the interval will be erroneous due to the switch in coefficients. Thus, the analysis proves that exact reconstruction can be achieved when the filters are changed at arbitrary intervals provided that the switch points are separated by a number of samples equal to or greater than the numerator order of the associated analysis polyphase filter. Assuming $n_i$ and $n_{i+1}$ are consecutive switch point indices in $n$, the criterion for exact reconstruction is

$$n_{i+1} - n_i \geq M_1.$$ 

We hasten to point out that this is an extremely mild restriction in practical applications. This is because the orders of practical recursive polyphase filters tend to be very low. Recursive filters which are first-order allpass polyphase filters can have magnitude characteristics comparable to a 24- or 32-tap QMF. Such filters have extremely good magnitude characteristics and generally
provide more than enough stopband rejection with sufficiently narrow transition bands. Because the polyphase filter order is just one in this case, the polyphase filters can be switched every other sample if needed.

4.2 Stability Issue

Another issue to address is stability. It is well known that if the filter coefficients are allowed to change arbitrarily, the system is not guaranteed to be stable. In our case, however, this is not a problem. In the formulation for the recursive time-varying analysis-synthesis system, the general problem is decomposed into piecewise constant coefficient system components, each of which is stable. Since the switch points are separated by $M_1$ samples, each region (after a switch) that is reconstructed by equation (26) can be represented uniquely by a unilateral $z$-transform term. In fact, all reconstructed regions that lie between switch points can be represented by separate unilateral $z$-transforms.

To illustrate this, consider the first-order allpass polyphase filter all-pass polyphase filters

$$P_0(z) = \frac{a - z^{-1}}{1 - az^{-1}}, \quad |a| < 1,$$

which we assume to be a filter employed over a finite interval. The corresponding difference equation is

$$y_0[n] = ay_0[n-1] + ax_0[n] - x_0[n-1]$$

and is used to generate all the samples in the interval. The initial conditions for the equation are derived from the output in the previous interval and in general will not be zero. If we take unilateral $z$-transforms on the above equation, we obtain the transform-domain equation

$$Y_0(z) = \frac{a - z^{-1}}{1 - az^{-1}} \left[ X_0(z) + \frac{ay_0[-1] - x_0[-1]}{a - z^{-1}} \right]$$

where $y_0[-1]$ is the initial condition. Since, the value of $a$ is less than one, the equation output will always be bounded.

The stability is thus completely determined by the locations of the poles of the corresponding transform. That is to say, for stability the poles associated with the analysis polyphase filters should be inside the unit circle and the zeros of the analysis polyphase filters (which become the poles of the synthesis polyphase filters) should be outside the circle.

5 Experiments and Performance Analysis

The analysis presented here used the simple case of a two-band polyphase system to develop the reconstruction equation derivation. Since the result applies to the actual polyphase and inverse polyphase filtering operations, banks with complex-valued channels [11] and an infinite variety of tree structures composed of two-band and/or parallel form filter banks.
Finally, as alluded to in the introduction, time-varying filter banks can improve the subjective and objective performance because of the input dependency of the error characteristics. This is clearly the motivation for the development of time-varying filter banks. It is also a measure by which one can judge its utility.

As an initial gauge of performance, we compare a conventional two-band subband coder and a two-band subband coder with time-varying filter coefficients. A finite length sequence was used as the input — shown in Figure 7 as the solid line. This input is a section of a row taken from the 256 x 256 test image “Lena.” In each case, the lowpass channel was coded with 4-bit uniform quantization while the highpass channel was coded with 2 bits. Notice that the conventional subband coder displays ringing (or overshoot-undershoot) distortion at the discontinuities as indicated by the dashed line in Figure 7. However, the other regions of the row are represented well. The magnitude and step response characteristics for $H_0(z)$ used in the conventional subband coder are shown in Figure 8.

The new subband coder with time-varying coefficients employs two filter sets, one with good magnitude response characteristics (i.e. $H_0(z)$ shown in Figure 8) and the other with good step response characteristics. The magnitude and step response characteristics for the latter filter are shown in Figure 9. In the regions of large discontinuity, the filter set with good step response characteristics is used. In other regions, the filter set with good magnitude response characteristics is used. Figure 10 shows the improvement that results for this example.
Figure 9: Magnitude and step response for the second lowpass filter used in the time-varying filter bank.

Figure 10: Performance of the new subband coder with time-varying coefficients. The solid line shows an image row taken from “Lena.” The dashed line is the coded result.
6 Remarks

Such an approach involves having to convey switching point information to the receiver for reconstruction. This can be done simply by sending side information as was the case in the example or potentially through a procedure based on feedback, the latter of which is presently being studied. For natural images, the amount of side information is negligible. Thus, even with the transmission of side information, this new approach seems attractive.
References


Appendix: Derivation of Shift Property

The unilateral z-transform is defined as

\[
X_R(z) = \sum_{n=0}^{\infty} x[n]z^{-n}.
\]

If \(x[n]\) is shifted by a positive integer time shift \(N\), and the transform computed, we obtain

\[
\sum_{n=0}^{\infty} x[n - N]z^{-n}.
\]

Substituting \(n = k + N\), results in

\[
\sum_{k=-N}^{\infty} x[k]z^{-k-N} = z^{-N} \sum_{k=0}^{\infty} x[k]z^{-k} + z^{-N} \sum_{k=-N}^{-1} x[k]z^{-k} = X_R(z)z^{-N} + z^{-N} \sum_{k=-N}^{-1} x[k]z^{-k}.
\]

If we let \(k = N - \ell\) and substitute, we obtain

\[
X_R(z)z^{-N} + \sum_{\ell=0}^{N-1} x[\ell - N]z^{-\ell}.
\]

Because this is valid only for integer \(N > 0\) we can append the term \(u[-N-1]\). For negative shifts, \(N < 0\), the transform becomes

\[
\sum_{n=0}^{\infty} x[n - N]z^{-n} = \sum_{k=-N}^{\infty} x[k]z^{-k-N} = z^{-N} \sum_{k=0}^{\infty} x[k]z^{-k} - z^{-N} \sum_{k=0}^{-N-1} x[k]z^{-k} = X_R(z)z^{-N} - z^{-N} \sum_{k=0}^{-N-1} x[k]z^{-k}u[-N-1]
\]

where \(u[-N-1]\) restricts \(N\) to be less than zero. Combining these results yields

\[
x[n - N] \iff z^{-N}X_R(z) + z^{-N}\sum_{\ell=0}^{N-1} x[\ell - N]z^{-\ell}u[N - 1] - z^{-N}\sum_{k=0}^{-N-1} x[k]z^{-k}u[-N - 1].
\]

For the anti-causal case, the derivation is similar. The transform of the time shifted input is

\[
\sum_{n=\infty}^{-1} x[n - N]z^{-n}.
\]

Assuming \(N > 0\) and substituting \(n = k + N\), results in

\[
\sum_{k=-\infty}^{-N-1} x[k]z^{-k-N} = z^{-N} \sum_{k=-\infty}^{-1} x[k]z^{-k} - z^{-N} \sum_{k=-N}^{-1} x[k]z^{-k} = X_L(z)z^{-N} - z^{-N}\sum_{k=1}^{N} x[-k]z^{k}
\]
For $N < 0$, the transform becomes

$$
\sum_{n=-\infty}^{-1} x[n-N]z^{-n} = \sum_{k=-\infty}^{-N-1} x[k]z^{-k-N} = z^{-N} \sum_{k=-\infty}^{-1} x[k]z^{-k} + z^{-N} \sum_{k=0}^{-N-1} x[k]z^{-k}
$$

Combining both results we obtain

$$
x[n-N] \iff z^{-N}X_L(z) - z^{-N} \sum_{\ell=1}^{N} x[-\ell]z^{\ell}u[N-1] + z^{-N} \sum_{\ell=0}^{-N-1} x[\ell]z^{-\ell}u[-N-1]. \quad (27)
$$
SPATIALLY-VARYING IIR FILTER BANKS FOR IMAGE CODING

Wilson C. Chung and Mark J. T. Smith

School of Electrical Engineering
Georgia Institute of Technology
Atlanta, GA 30332

ABSTRACT
This paper reports on the application of spatially variant IIR filter banks to subband image coding. The new filter bank is based on computationally efficient recursive polyphase decompositions that dynamically change in response to the input signal. In the absence of quantization, reconstruction can be made exact. However, by proper choice of an adaptation scheme, we show that subband image coding based on time varying filter banks can yield improvement over the use of conventional filter banks.

1. INTRODUCTION
Recursive analysis and synthesis filter banks for subband coding have been considered previously and were shown to be particularly attractive because they preserve exact reconstruction and can achieve tremendous computational efficiency relative to comparable FIR systems [6], [8]. These filter banks are typically two-band systems that serve as building blocks for uniform and non-uniform band tree-structured systems.

Much attention has been given in minimizing the perceptually significant coding distortions in image coding systems. At low rates, aliasing and ringing distortions tend to be pronounced and have spurred some efforts to reduce these effects; but the success has been limited [9], [6]. The difficulty is that the step response ripples of the filters, which are the source of the ringing distortion, are needed to achieve good filter magnitude response characteristics. Good spectral magnitude characteristics are important for reducing the effects of aliasing in the presence of coding. Aliasing distortion appears as blurring in the image and is subjectively noticeable. If filters with good magnitude characteristics are used to reduce the aliasing, then ringing distortion results which is also very noticeable and objectionable. Since both good step response and good magnitude response characteristics cannot be achieved simultaneously for conventional filter banks, this poses a dilemma.

A solution to this problem using IIR time-varying filter banks was proposed at the IEEE DSP Workshop [2] and a theoretical analysis of the filter bank was presented in [4]. In this paper, we expand upon this idea giving special attention to its application to image coding. In the next section, we briefly summarize the results derived in [4] which are the basis for this work. In the follow section we consider a control mechanism for performing the adaptation and compare the performance of the overall system to that of a conventional system.

2. SPATIALLY-VARIANT FILTER BANKS
The two-band polyphase system, as shown in Figure 1, is commonly used in conventional subband analysis/synthesis systems. The lowpass and highpass analysis filters, \( H_0(z) \) and \( H_1(z) \), have the form:

\[
H_0(z) = P_0(z^2) + z^{-1}P_1(z^2) \tag{1}
\]

\[
H_1(z) = P_0(z^2) - z^{-1}P_1(z^2) \tag{2}
\]

where \( P_0(z) \) and \( P_1(z) \) are the analysis polyphase filters and \( Q_0(z) \) and \( Q_1(z) \) are the synthesis polyphase filters. This system has the property that its structure guarantees exact

This work is supported in part by the National Science Foundation under contract MIP-9116113 and the National Aeronautics and Space Administration.
reconstruction when
\[ Q_0(z) = \frac{1}{P_0(z)} \quad \text{and} \quad Q_1(z) = \frac{1}{P_1(z)}. \]
The polyphase filters, \( P_0(z) \) and \( P_1(z) \), are assumed to be causal stable recursive filters with zeros strictly outside the unit circle and have allpass or near allpass characteristics. The synthesis polyphase filters, \( Q_0(z) \) and \( Q_1(z) \), by definition will have their poles outside the unit circle and their zeros inside the circle. Consequently they are anti-causal and stable.

The new spatially-varying system is identical except that the filters change in response to the input. In particular, the coefficients of the polyphase filters \( P_0(z) \) and \( P_1(z) \) are changed selectively at some point in the filtering process to another set of polyphase filters. Thus to preserve exact reconstruction it is sufficient to guarantee that \( v_i[n] = v_i[n] \) (where \( i = 0, 1 \)) at all times before, during, and after the switching of coefficients. The derivation presented in [4] considered starting with a set of polyphase filters \( (p_{0}^L[n], \ p_{L}^R[n]) \), and switching them to another set of filters \( (p_{0}^l[n], \ p_{L}^r[n]) \). For convenience, assume that we switch the filters at \( n = 0 \) and that the polyphase filters have the form:
\[
\begin{align*}
p^L(z) &= \frac{a_0^L + a_1^L z^{-1} + \ldots + a_{M_0}^L z^{-M_0}}{1 + \beta_1^L z^{-1} + \beta_2^L z^{-2} + \ldots + \beta_{N_0}^L z^{-N_0}}, \\
p^R(z) &= \frac{a_0^R + a_1^R z^{-1} + \ldots + a_{M_1}^R z^{-M_1}}{1 + \beta_1^R z^{-1} + \beta_2^R z^{-2} + \ldots + \beta_{N_1}^R z^{-N_1}}.
\end{align*}
\]
The labels \( L \) and \( R \) signify that the signal or filter coefficient is associated with the left and right half of the time index respectively.

The direct form difference equations that implement the polyphase filters are:
\[
\begin{align*}
v_L[n] &= \sum_{m=0}^{M_0} a_m^L v[n - m] - \sum_{\ell=1}^{N_0} \beta_{\ell}^L v_L[n - \ell], \\
v_R[n] &= \sum_{m=0}^{M_1} a_m^R v[n - m] - \sum_{\ell=1}^{N_1} \beta_{\ell}^R v_R[n - \ell]
\end{align*}
\] in the range \(-\infty < n \leq -1 \)
and
\[
\begin{align*}
v_L[n] &= \sum_{m=0}^{M_0} a_m^L v[n - m] - \sum_{\ell=1}^{N_0} \beta_{\ell}^L v_L[n - \ell], \\
v_R[n] &= \sum_{m=0}^{M_1} a_m^R v[n - m] - \sum_{\ell=1}^{N_1} \beta_{\ell}^R v_R[n - \ell]
\end{align*}
\] when \( 0 \leq n < \infty \). Note that \( v_L[n] \) and \( v_R[n] \) are the left-sided and right-sided components of a polyphase filter output \( v[n] \), i.e. \( v[n] = v_L[n] + v_R[n] \). Equation (4) requires the initial conditions \( v_L[-N_1], v_R[-N_1 - 1], \ldots, v_R[-1] \) in order to evaluate \( v_L[0] \). These initial conditions are obtained from the past samples of the output, i.e. \( v_L[n] = v_L[n] \) in the range \(-N_1 \leq n \leq -1 \).

The reconstruction equations (which are derived in [4]) are performed by anti-causal filtering and are defined by the difference equations
\[
\begin{align*}
x_L[n - M_0] &= \frac{1}{\alpha_{M_0}^L} \sum_{\ell=1}^{M_0} a_{\ell}^L x_L[n - \ell - M_0] + \\
\frac{1}{\alpha_{N_0}^L} \sum_{m=0}^{N_0} \beta_{\ell}^L v_L[n + m - N_0]
\end{align*}
\] for \( 0 \leq n < \infty \) and
\[
\begin{align*}
x_R[n - M_1] &= \frac{1}{\alpha_{M_1}^R} \sum_{\ell=1}^{M_1} a_{\ell}^R x_R[n + \ell - M_1] + \\
\frac{1}{\alpha_{N_1}^R} \sum_{m=0}^{N_1} \beta_{\ell}^R v_R[n + m - N_1]
\end{align*}
\] for \(-\infty < n \leq -1 \). The assumption that the switching occurs at \( n = 0 \) was done purely for convenience. But clearly these equations hold regardless of when the filters are switched. However, there is a constraint on how close together we can switch our filter. Switching points should occur at least \( N_0 \) samples apart. In the next section, we consider how to determine when to switch a filter and what characteristics the filters should have in order to improve subjective and objective performance.

3. ADAPTATION CONTROL SYSTEM

The flexibility to switch among a set of different filters is potentially very powerful. But the key to make this work well is determining an effective control system for adapting the filters relative to the signal. Obviously there are many possibilities. The one investigated in this work is based on the observation that filters with good step response characteristics tend to perform better in local regions with sharp edges or transitions. Filters with good magnitude response characteristics seem to be better suited for the other regions. Thus two sets of filters are used in this system: filter set \( A \), which has good magnitude response characteristics; and filter set \( B \), which has good step response characteristics.

For example, for the test image Lena, only about 0.0281
bits/pixel was consumed by this side information. If, on the other hand, \(y_0[n]\) is used as an estimation point, sending side information can be avoided. More precisely, the estimation should be performed on \(y[n]\) after it has been coded. This allows the synthesis section to derive the switching point from the exact same signal used by the analysis section to derive the switching point. This form of the implementation is a feedback adaptation approach. Interestingly, there is often enough information in the \(y_0[n]\) signal to make an accurate estimation of major transitions. Even if the estimation is in error occasionally, the results are not catastrophic since the system performance is lower bounded by that of conventional subband coding (which is not bad).

4. SIMULATION RESULTS

To evaluate the new analysis/synthesis system, we considered a baseline pyramid two-level system with seven subbands and the Enumerative Laplacian Quantization (ELQ) encoder introduced by Darragh [10]. Further, we used the method of circular convolution to periodically extend our image boundaries [6]. The derivation of initial conditions for 2-D exact reconstruction with IIR filters is addressed in [6]. The 256 × 256 test image Lena was used in this example. A section of the original is shown in Figure 4. Figure 5 shows Lena coded at 0.4527 bits/pixel with conventional IIR filter banks. The filters used are recursive (filter set A) with magnitude response and step response plots shown in Figures 3 and 2. The output corresponding to the new system is shown in Figure 6. It is coded at the same bit rate and uses \(y_0[n]\) as an estimation point. Approximately 0.0281 bits/pixel are devoted to the side information associated with the adaptation. In our feedforward system, we use an arithmetic coder as a lossless coder for the side information. The subjective quality is noticeably better as shown by examining both coded outputs. The ringing distortion is quite noticeable around the boundary of the hat, as seen in 5. On the other hand, the ringing distortion disappeared in the coded image using the new system, as seen in 6. Little magnitude distortion can be seen around the edges since the filter set of good step response and bad magnitude response is allowed to switch back very quickly after only a few samples. This improvement in quality is also reflected in the PSNR of the coded results.

Several important aspects should be addressed in optimizing the performance of the new coder. First, the intelligent selection of threshold levels in determining the edges or boundaries within an image is critical. A good edge or level detector almost surely guarantees the success of the adaptation control system. Secondly, the issue of signal extension should be study carefully in light of the new spatially variant system because the derivation of initial conditions no longer depends on only one distinct filter set, but many. Finally, a feedback scheme can be considered where side information for edge locations within an image will no longer be needed a priori for transmission over the channel. As with most image coding strategies, much of the improvement in quality comes as a result of fine tuning the coder. This has not be done yet at this point but should be done in time for presentation at the conference.
5. REFERENCES


Finite-State Residual Vector Quantization*

Faouzi Kossentini and Mark J. T. Smith
Digital Signal Processing Laboratory
School of Electrical Engineering
Georgia Institute of Technology
Atlanta, GA 30332

Christopher F. Barnes
Georgia Tech Research Institute
Georgia Institute of Technology
Atlanta, GA 30332

June 24, 1992

*This work was supported by the National Science Foundation under contract MIP-9116113.
Finite-State Residual Vector Quantization

Mark J. T. Smith
Digital Signal Processing Laboratory
School of Electrical Engineering
Georgia Institute of Technology
Atlanta, GA 30332
(404) 894-6291
(404) 894-8363 (FAX)
June 24, 1992

Abstract

This paper introduces a new class of vector quantizers (VQ) and discusses their application to image compression. In simplest terms, the new VQ may be viewed as a merger of residual vector quantization (RVQ) and finite state VQ (FSVQ). Experimental results are presented that illustrate the magnitude of improvement attainable over either RVQ or FSVQ individually.

Unique to this paper is the introduction of a new design algorithm that jointly optimizes the RVQ stage codebooks with multi-path search capacity and variable bit rate capability. The resulting improvement in reproduction quality exceeds that of both the sequentially optimized [33] and jointly optimized codebooks [4] considered previously. Furthermore, the new VQ requires only a fraction of the codebook memory needed for conventional FSVQ and, in addition, has high computational efficiency.
1 Introduction

Vector Quantization (VQ) has recently received much attention as a powerful technique for data compression [16, 33], particularly for coding images and motion video at low bit rates [25], [34]. Many excellent references are available on this general topic [3], [14], [16], [17].

A motivation for the use of VQ is the information theoretic advantage obtained by using vectors in place of scalars. Well-known results in rate distortion theory [6] state that the minimum bit rate for a source with a given fidelity requirement can be approached by increasing the block (or vector) size. The superiority of VQ over scalar quantization is mainly due to its ability to better exploit the statistics of a random process. Unlike scalar PCM, VQ can exploit both the linear and non-linear dependencies that may exist within an input vector. Even if the random vectors are statistically independent, VQ can still take advantage of its high dimensionality to exercise freedom in choosing cell geometries, leading to better compaction of the input space.

For all its benefits, VQ is not without its limitations. Rate distortion theory, on the one hand, calls for the use of large vectors while costs associated with encoding complexity and memory requirements encourage the use of small vectors. Thus as a compromise, relatively small vector sizes, typically of size $4 \times 4$, are usually used in practical VQ systems. Although the VQs used in practice fall short of reaching their performance potential, they still perform better than scalar quantizers.

While conventional VQ exploits the dependencies among the pixels in the vector, it does nothing to take advantage of the inter-dependencies among vectors. To capture the additional information in the surrounding pixels, the vector size must be increased or the correlation between successive blocks must be exploited. In the case of the former, constrained codebooks with memory efficient structure such as RVQ can be employed. In the case of the latter, predictive schemes such as FSVQ have been shown to be effective. Systems based on the combination of both techniques (called FS-RVQ)
were designed, by the authors of this paper, using 8 x 8 vectors, and were shown [29] to achieve better quality than memoryless VQ with only a fraction of the memory and computation. These residual VQ and finite state ideas are further developed and extended in this paper to include variable rate coding and multi-path codebook search capability, both of which are incorporated in the VQ design algorithm. The new class of VQ that emerges is called variable-rate finite-state VQ or VR-FS-RVQ.

To set the notation and terminology used throughout this paper, the next section begins with a brief recap of VQ preliminaries. To lay the foundation for the discussion of residual VQ and the development of the new design algorithm, classical design methods and constrained VQ systems are also discussed in the section as well. Several variations of the RVQ system are derived and contrasted after which their union with finite state prediction is considered in section 5. FS-RVQ and variable-rate FS-RVQ are formally treated in the latter sections and their performance results are compared. There it is shown that important gains can be made on several fronts: first in terms of performance relative to conventional FSVQ; second in terms of computational complexity, relative to exhaustive search VQ; third in terms of memory, relative to conventional VQ, FSVQ, and tree-search VQ; and finally in terms of bit rate variability.

2 Vector Quantization

As is well-known, vector quantization is the extension of scalar quantization to higher dimensional spaces. It is essentially a pattern matching procedure that codes entities called vectors as illustrated in Figure 1. More precisely, a vector quantizer \( Q \) of dimension \( k \) and size \( N \) is a mapping from \( k \)-dimensional Euclidean space, \( \mathbb{R}^k \), into a finite set \( C \subset \mathbb{R}^k \) containing \( N \) output or reproduction points, called code vectors. Thus,

\[
Q : \mathbb{R}^k \rightarrow C,
\]

(1)
where $C$ is called the codebook and has size $N$, or equivalently, $N$ distinct code vectors $\mathbf{y}_i \in \mathbb{R}^k$, $1 \leq i \leq N$.

Associated with every $N$ point vector quantizer is a partition of $\mathbb{R}^k$ into regions or cells, $V_i$, $1 \leq i \leq N$. The $i$th cell is defined by

$$V_i = \{ \mathbf{x} \in \mathbb{R}^k : Q(\mathbf{x}) = \mathbf{y}_i \}. \tag{2}$$

The number of bits per pixel, $r$, used to address the codebook is

$$r = (\log_2 N)/k \tag{3}$$

and the minimum average bit rate for encoding the input, known as the entropy of the quantizer $Q(\cdot)$ [6], is:

$$h = - \sum_{i=1}^{N} p(\mathbf{x} \in V_i) \log_2 p(\mathbf{x} \in V_i) \tag{4}$$

where $h$ is expressed in bits per vector. For discrete amplitude sources, the entropy is a positive quantity bounded above by the product $kr$, i.e.

$$0 \leq h \leq \log_2 N = kr \tag{5}$$

When a vector $\mathbf{x}$ is quantized as $\mathbf{y}$, a quantization error generally results and a distortion measure $d(\mathbf{x}, \mathbf{y})$ can be defined between $\mathbf{x}$ and $\mathbf{y}$. The statistical average of the distortion for a vector quantizer $Q(\cdot)$, can be written as

$$D = E\{d(\mathbf{x}, Q(\mathbf{x}))\} = \int d(\mathbf{x}, Q(\mathbf{x})) f(\mathbf{x})d\mathbf{x} \tag{6}$$

where $f(\mathbf{x})$ is the probability density function (pdf) of the input.

A vector quantizer is said to be an optimal (minimum distortion) quantizer if the distortion in (6) is minimized over all $N$-level vector quantizers. As in scalar quantization, there are two necessary conditions for optimality of vector quantizers. The first condition is that the optimal VQ is obtained using a minimum-distortion or nearest-neighbor rule:

$$Q(\mathbf{x}) = \mathbf{y}_i \quad \text{iff} \quad d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j), \quad i \neq j, \quad 1 \leq j \leq N. \tag{7}$$
That is, the vector quantizer chooses the code vector $y_i$ that results in the minimum distortion with respect to $x$, with ties being broken arbitrarily. The second necessary condition for VQ optimality is that each code vector $y_i$ is chosen so as to minimize the average distortion given a partition cell. That is, $y_i$ is the vector $y$ which minimizes the conditional distortion

$$ D_i = E\{d(x,y) | x \in V_i\} = \int_{x \in V_i} d(x,y) f(x) dx. \quad (8) $$

That vector $y_i$ is called the centroid of the cell $V_i$. The computation of the centroid for a particular cell depends on the distortion measure $d(\cdot, \cdot)$. The most popular distortion measure is the mean squared error defined by

$$ d(x, \hat{x}) = ||x - \hat{x}||^2 = \sum_{i=1}^{k} (x_i - \hat{x}_i)^2, \quad (9) $$

where $|| \cdot ||$ denotes the Euclidean norm and $x_i$ and $\hat{x}_i$ are elements of the vectors $x$ and $\hat{x}$ respectively. The mean squared error is used very frequently in VQ design partly because of its long history, its mathematically tractable nature, and its ability to handle input-dependent weightings.

Both the nearest-neighbor condition and the centroid condition are generalizations of the ones derived for optimality of scalar quantizers. These conditions are important for they are frequently used as the basis for many VQ design algorithms.

2.1 VQ Design Algorithms

The goal of the VQ codebook design is to find a set of code vectors that minimizes the overall average distortion in (6) at a given bit rate. Since the probability distributions of many practical sources such as images are not known, the codebook design process is usually based on a large training set (of size $L$) where the training vectors are taken from the source to be encoded. If the input process is stationary and ergodic, then sample averages converge to expectations, i.e., the average distortion

$$ D = \frac{1}{L} \sum_{n=1}^{L} d(x_n, Q(x_n)) \quad (10) $$
converges (as \( L \to \infty \)) to the expected value given in equation (6). In practice most processes are neither stationary nor ergodic. However, it is often reasonable to assume that they are in analysis, and statistical averages are replaced by time averages during the design process [17].

The most popular VQ codebook design technique is the generalized Lloyd algorithm (GLA) or LBG algorithm after [32]. It uses a set of statistically representative samples of the source output (i.e. a training set) to iteratively design a codebook. The LBG algorithm is based on simultaneously satisfying the necessary conditions for optimality given by equations (7) and (8), and does not have the ability to locate an optimal codebook [15]. Thus, several techniques such as simulated annealing [44] and stochastic relaxation [45] have been considered to help force convergence to a globally optimal code book. Iterative design methods of this type tend to be very computationally intensive. Non-iterative design techniques such as the pairwise nearest neighbor (PNN) algorithm [11], and the self-organizing feature map (SOFM) algorithm [42, 35], have also been considered and can be used to design codebooks in a fraction of the time required by the LBG. They are not guaranteed to converge to local optima like the LBG algorithm but experimental results indicate reasonably comparable performance results when applied to image coding [43].

2.2 VQ Issues and Constraints

For conventional unconstrained VQ, as just described, the best matching code word for a given input vector is obtained by exhaustively searching the VQ codebook. Thus this kind of VQ is often called exhaustive-search VQ or ESVQ.

An issue of recognized importance for ESVQ is that the size of the codebook grows exponentially as a function of both vector size \( k \) and bit rate \( r \). An ESVQ codebook \( C \) consisting of \( N \) code vectors (each of dimension \( k \)), requires

\[
N = 2^k
\]  

(11)

vector distortion calculations to quantize an input vector \( x \) as seen from Equation

6
(3). Similarly, the memory required to store the codebook at both the encoder and decoder grows exponentially and is given by \( kN \) pixels, or equivalently, \( k2^r_k \) pixels.

To offset these complexity and memory problems, various imposed structural constraints have been considered by many authors. These constraints can generally be expected to lead to reduced performance for a given rate and dimension. However, the reduction in complexity and/or memory obtained usually more than compensates for the loss in quality. Some typical examples of structured vector quantizers are product code VQ [38, 7], lattice VQ [9], hierarchical VQ [40], and tree-searched VQ (TSVQ) [8, 16]. Residual Vector Quantization (RVQ), a special class of product code VQ, is a technique whose structure reduces both the memory and computation costs [24]. It too trades memory and computational efficiency for performance quality, but by employing jointly optimized design algorithms (to be discussed later), the quality loss can be minimized.

3 Residual Vector Quantizers

Residual vector quantization (RVQ), often called multistage VQ, is a method intended to reduce the VQ computation and memory requirements. It consists of a cascade of VQ stages, each operating on the "residual" of the previous stage. A block diagram of a two-stage RVQ is given in Figure 2 for illustration. The input vector \( x^1 \) is first quantized using the vector quantizer \( Q^1 \). The residual vector \( x^2 \) is then obtained by subtracting the quantized vector \( \hat{x}^1 \) from the original vector \( x^1 \). This residual is then used as the input to a second vector quantizer \( Q^2 \) which produces the quantized output \( \hat{x}^2 \). The final quantized value of \( x^1 \) is simply the sum of the two vectors \( \hat{x}^1 \) and \( \hat{x}^2 \). A general RVQ consisting of \( P \) stages is capable of uniquely representing \( N = \prod_{i=1}^{P} N_i \) vectors with only \( \sum_{i=1}^{P} N_i \) code vectors required for storage. Thus, the RVQ achieves tremendous savings over unconstrained VQ in terms of memory requirements. Of special interest is the binary RVQ (BRVQ) shown in Figure 3. BRVQ is most efficient in terms of memory and may be associated with the binary
tree structure shown in Figure 4. BRVQ is the most constrained member of the RVQ family and consequently suffers the most quality loss in general. A wide variety of RVQ's are available based on using larger numbers of vectors in the stages. As the number of vectors per stage is increased, so does the performance. This point is illustrated in the plots shown in Figure 5.

The first introduction of the RVQ structure was by Juang and Gray [24] for the coding speech signals. Their design procedure consisted of sequentially applying the LBG algorithm to the residual training sets. In their speech coding experiments, a slight loss in performance was reported relative to unconstrained VQ when using a two-stage VQ. In later work, Makhoul, Roucos, and Gish [33] performed experiments using multistage VQ, and reported that its performance degrades when the number of stages increases.

Some of the disparity in performance can be attributed to two factors. First the decoder is constrained to have a direct sum codebook. In other words, all possible code vectors reachable by the RVQ, are formed by the sum of stage code vectors—the set of which we call the equivalent codebook. Second, entanglements—which are often discussed in the context of tree-search VQ—tend to complicate the task of implementing stage partitions [4]. However, some of the degradation observed in the Juang and Gray design procedure can be attributed to a miss-match between the design rules used to generate the codebook at each stage and the inherent structure of the RVQ. In [4], a new RVQ analysis and design was introduced and shown to yield improvement over its predecessors. Building on this work, an algorithm is introduced here that demonstrates further improvement.

3.1 The New RVQ Design Algorithm

The RVQ design procedure, introduced in [4], attempts to jointly optimize each stage codebook to minimize the reconstruction error over all training data subject to the structural constraints imposed by the RVQ framework. It is very similar to the LBG algorithm in the sense that each iteration tries to simultaneously satisfy both the
centroid condition and the nearest-neighbor condition.

Let $x^1$ be an input vector described by a probability density function (pdf) $f$ on $\mathbb{R}^k$. A $P$-stage RVQ consists of a finite sequence of $P$ stages such that the first stage quantizes the source vector $x^1$ and the $p$th stage quantizes the residual vector $x^p$. The sequence of quantizer triples \{$(C^p, Q^p, \mathcal{P}^p)$; $1 \leq p \leq P$\} describes a $P$-stage RVQ. The code vectors comprising the stage codebook $C^p$ and the cells comprising the partition $\mathcal{P}^p$ are indexed with the subscripts $j^p$, where $j^p$ is the $p$th stage code vector index. Also, let the distortion that results from representing $x^1$ with $\hat{x}^1$ be expressed by $d(x^1, \hat{x}^1)$ and its average be

$$D(x^1, \hat{x}^1) = E\{d(x^1, \hat{x}^1)\}.$$ \hfill (12)

A $P$-stage RVQ is optimal for the source described by the pdf $f$ if it gives at least a locally minimum value of the average distortion. However, this minimization is complicated by the fact that $D(x^1, \hat{x}^1)$ requires knowledge of the joint pdf $f(x^1, x^2, ..., x^P)$, which depends in a complicated fashion upon the sequence of stage codebooks and partition cells.

The minimization problem can be made much easier by minimizing the average distortion introduced by the equivalent single-stage quantizer. The equivalent VQ and RVQ are identical in the sense that they produce the same representation of the source output, and they have the same expected average of distortion. The equivalent quantizer can be represented by the triple $(C^e, Q^e, \mathcal{P}^e)$, where $C^e$ contain all ordered sums of the code vectors in the stage codebooks, $Q^e$ is the quantizer mapping defined by

$$Q^e(x^1) = \sum_{p=1}^{P} Q^p(x^p)$$ \hfill (13)

and $\mathcal{P}^e$ is the equivalent partition of the input space. The equivalent code vectors comprising the equivalent codebook $C^e$ and the cells comprising the equivalent partition $\mathcal{P}^e$ are indexed with the subscripts $j^e$, where $j^e$ is the equivalent code vector index. As illustrated in Figures 3 and 4, the $j^e$th equivalent code vector is represented by the $j^e$th path in the RVQ tree.
For convenience, the concept of a *graft residual vector*, $\xi^p$, is introduced and defined as the difference between $x^1$ and the sum of all stage code vectors in the *grafted residual path*. A grafted residual path is the path obtained by removing the $p$th node and connecting the resulting two branches. Thus, for each possible input vector $x^1 \in \mathbb{R}^k$, the $p$th stage graft residual vector, $\xi^p$, is defined as

$$\xi^p = x^1 - \sum_{i=1}^{P} Q^i(x^i) \quad (14)$$

For an RVQ to have minimum average distortion, the multistage code vectors $y^p_j$ at the $p$th stage must satisfy [4]

$$E \left\{ d \left( \xi^p, y^p_j \right) \mid x^p \in S^p_j \right\} = \min_{u \in \mathbb{R}^k} E \left\{ d \left( \xi^p, u \right) \mid x^p \in S^p_j \right\} \quad (15)$$

where $S^p_j$ is the $j$th cell of the partition $P^p$. The $y^p_j$'s which satisfy (15) are generalized centroids of residuals formed from the encoding decisions of all stages prior and subsequent to the $p$th stage.

Equation (15) forms the basis for the decoder optimization step of the RVQ design algorithm. The joint optimization of stage codebooks can be carried out either simultaneously (joint update) or successively for each stage (sequential update). Here, the stage codebooks are updated sequentially.

For jointly optimal partitioning, the RVQ encoder must map the input vector $x^1$ according to the rule [4]

$$x^1 \in S^e_j \iff d(x^1, y^e_j) \leq d(x^1, y^e_{j^*}) \quad \forall \ k^e \neq j^e \ 1 \leq k^e \leq N^e \quad (16)$$

where $S^e_j$ is the $j$th cell of the partition $P^e$ and $y^e_j$ is the $j$th equivalent code vector.

Equation (16) is the basis of the encoder optimization step of the new RVQ algorithm. It involves the mapping of input vectors using a nearest-neighbor rule with respect to the set of all equivalent code vectors. This nearest-neighbor condition is identical to the one derived in [32] for conventional VQ, and can be satisfied by *exhaustive search* RVQ (ES-RVQ) encoders. These encoders effectively compare the
input vector with each code vector in the equivalent codebook. They encoders are capable of finding the best match for an input vector but are very computationally demanding.

At the other extreme is the sequential search RVQ or SS-RVQ encoder. It consists of successively selecting the best matching vector at each stage in the RVQ and therefore has very fast search capability. The price paid for the speed is a loss in performance since there is no longer a guarantee of obtaining the optimal code vector in the equivalent codebook.

Since the RVQ has a tree structure, efficient search techniques such as multipath searching, which were developed for tree encoding, can be used to search the RVQ codebooks. The gap between ES-RVQ and SS-RVQ can be bridged by using multipath-search or $M$-search to more closely satisfy the nearest-neighbor mapping rule without exhaustively searching all the paths in the RVQ tree [30]. The $M$-search technique, introduced in [23], was shown to be very efficient when used to search the RVQ tree [30].

The $M$-search algorithm [23] proceeds one level deeper into the RVQ tree by extending all branches from the $M$ saved nodes, and only the best $M$ of these branches are saved for the next level. This procedure continues until the last stage of the codebook is reached, and then the code vector of the best path among the final $M$ saved paths is used. The $M$-search RVQ design algorithm employs $M$-search in the encoding step, and the computational costs are then increased by a factor of $M$ (approximately). Table 1 shows memory, encoder complexity, decoder complexity and performance of ESVQ, TSVQ, ES-RVQ, SS-RVQ and MS-RVQ. In [30], the $M$-search RVQ design algorithm was shown to trade a relatively moderate increase in complexity for a significant improvement in performance.

### 3.2 Comments on Algorithm Performance

Several experiments were made to evaluate the complexity and performance of RVQ using the new design algorithm. This was done in the context of coding 8-bit
monochrome images of size 512 × 512. In the first set of experiments, the performance of codebooks designed using the SS-RVQ algorithm is compared to that of codebooks designed using Juang and Gray's algorithm. Figure 6 shows the PSNR performance of both codebooks for the test image Lena when 4 × 4 and 8 × 8 vectors are used. The SS-RVQ algorithm tends to perform only slightly better at low bit rates. However, as the bit rate is increased, a significant improvement is observed (about 2 dB at 1 bpp). When M-search is employed during the encoding step of the RVQ design, a greater gain can be obtained (about 3 dB at 1 bpp), as shown in Figure 7.

In a second set of experiments, the complexity and performance of RVQ designed using the new design algorithm are compared to those of ESVQ. Table 2 shows the PSNR performance, entropy, codebook storage and encoding complexity of ESVQ and several RVQ's for the test image Lena at 0.625 bpp when 4 × 4 vectors are used. It is clearly seen that the performance of RVQ tends to approach that of ESVQ as \( N \) (number of vector per stage) and/or \( M \) (number of searching paths) is increased.

The use of entropy-based coders, such as Huffman and arithmetic coders, in conjunction with fixed rate systems (like VQ) has been shown to work well in many practical systems. As a final point of comparison, the performance of RVQ and VQ are compared with the assumption that its output symbols are entropy coded, results of which are shown in Table 2. The output entropy of the RVQ tends to be significantly lower than that of the unconstrained ESVQ. While the RVQ cannot fully exploit the dependencies within the vector (because of inherent constraints in the RVQ), it may have a large number of equivalent code vectors which have a very low probability of being used. In fact, experimental results (shown in Table 3) suggest that if the VQ system is to be followed by an entropy coder, the RVQ/entropy coder can perform nearly as well as a VQ/entropy coder but with much lower complexity and memory requirements.
4 Finite-State Vector Quantization

A number of predictive techniques have been proposed to incorporate memory into VQ system. One such technique is finite-state vector quantization (FSVQ), first proposed by Foster, Gray and Dunham in [12]. They introduced two equivalent FSVQ structures, the labeled-state and labeled-transition FSVQ, which are based on the Moore and Mealy machines [20], respectively. Here, the proposed FSVQ encoder resembles the labeled-transition FSVQ.

FSVQ can be viewed as a finite collection of vector quantizers where each successive input vector is encoded using a VQ codebook determined by the current encoder state. The current state and the transmitted channel symbol determine the next state. Early FSVQ systems were applied to speech [12] and later to images [1] and video [3]. In image coding, FSVQ exploits the two-dimensional correlation between image blocks, because natural images are often correlated well beyond the size of a typical vector block in both the horizontal and vertical directions. This allows for improvement in quality.

4.1 FSVQ Structure and Design

To lay the foundation for discussion of the new finite state approach, we first consider the conventional FSVQ structure and methods for designing such systems. Let $\mathbb{R}^k$ denote the $k$-dimensional Euclidean space and $\mathcal{S} = \{1, 2, \ldots, K\}$ denote the state space. A $k$-dimensional $K$-state FSVQ is specified by the state space $\mathcal{S}$, an initial state $s_0$, and the three mappings:

$$\alpha : \mathbb{R}^k \times \mathcal{S} \rightarrow \mathcal{N} \quad (17)$$
$$\beta : \mathcal{N} \times \mathcal{S} \rightarrow \mathcal{C} \quad (18)$$
$$\varphi : \mathcal{N} \times \mathcal{S} \rightarrow \mathcal{S} \quad (19)$$

where $\mathcal{N} = \{1, 2, \ldots, N\}$ denotes the $N$ alphabet channel symbols and $\mathcal{C}$ is the collection of all the reproduction codewords in all of the state codebooks. In other words,
Given an input source vector \( x \in \mathbb{R}^k \) and a state \( s \in S \), the first mapping, the finite-state encoder, encodes the vector \( x \) into the channel symbol or channel codeword \( \alpha(x, s) \in \mathcal{N} \). Also, given a channel symbol \( u \in \mathcal{N} \) and \( s \in S \), the second mapping, the finite-state decoder, produces a reproduction vector \( \hat{x} \in \mathcal{C} \). Finally, given a channel symbol \( u \in \mathcal{N} \) and \( s \in S \), the third mapping, the next state function, determines the next state. Both the encoder and the decoder must have knowledge of the current state. Since FSVQ is not permitted to send the side information usually used to specify the state, the decoder is forced to determine the current state from the previous state and the channel codeword. This implicit constraint implies that the decoder will be able to track the state sequence given a common initial state.

As before, a distortion measure \( d \) must be introduced. It assigns a non-negative cost \( d(x, \hat{x}) \) for the reproduction of the input vector \( x \) with the output vector \( \hat{x} \). Then the encoder is specified by the minimum distortion rule

\[
\alpha(x, s) = \arg \min_{u \in \mathcal{N}} d(x, \beta(u, s)), \forall s \in S.
\]  

The distortion measure \( d(\cdot, \cdot) \) is usually the mean squared error. For both speech and image sources, several perceptually-based distortion measures have been used [1, 16] previously and were shown to produce small performance improvements. The mean squared error, however, is still the most frequently used in FSVQ and is the one considered in this work.

A family of algorithms for the design of FSVQ's for waveform coding was first introduced in [12]. These algorithms consist of first designing an initial set of vector quantizers together with a next-state function setting the rule by which the next vector quantizer is selected. Standard codebook design algorithms, like the LBG, provide a means of optimizing the state codebooks and iteratively improving the encoder/decoder combination for a given next-state function. Note in this approach...
that although the state codebooks are being iteratively improved, the next state function remains invariant. The potential for improvement exists if the next-state function can also be optimized in the process. An alternate technique, introduced in [10], attempts to do this by iteratively improving the next-state function based on an algorithm from adaptive stochastic automata theory. Although this algorithm was shown to produce a stable code when applied to a speech source, no mathematical proof of its convergence has yet been developed.

Direct generalization of the previous FSVQ design techniques to image coding can lead to severe performance degradations due to the loss of local edge continuity across adjacent blocks. Aravind and Gersho [1] developed the FSVQ concept to make it effective for coding images by using a perceptually-based classifier (introduced in [36]) to define the states and the next state function. States are represented by a pair of classes, one class corresponding to the block directly above and the other class corresponding to the block to the left. Thus if there are $M$ classes, there will be $K = M^2$ states. This state determination rule is illustrated in Figure 8. The state $s_{i,j}$ for vector $x_{i,j}$ is defined by

$$s_{i,j} = \text{class}(\hat{x}_{i-1,j}), \text{class}(\hat{x}_{i,j-1})$$

where the "hat" symbol is used to denote that the vectors are coded. Since the coded vectors $\hat{x}_{i-1,j}$ and $\hat{x}_{i,j-1}$ are known to the decoder, no side information is needed to determine the next state. Other more general state configurations (shown in Figure 9) have been used [27, 2] and allow more local information to be incorporated in the finite state prediction. The disadvantage of using more of the surrounding blocks (for example $L$ of them) is that the total number of states (and state codebooks) becomes $M^L$, which is apt to be unmanageable. Thus using two adjacent blocks as shown in Figure 8 seems a reasonable compromise and is the one used in this work.

The choice of classifiers in [1] was influenced by the perceptual importance of preserving edge continuity and sharpness. Since the vector blocks are relatively small in comparison to the image size, it is assumed that only one edge will be present in a block and can be reasonably represented as a straight line bisection. Based on
this idea, the classifier distinguishes between four edge orientations (one horizontal, one vertical and two diagonal) and, for each orientation, between the two categories of black-to-white and white-to-black transition. In addition, the classifier recognizes blocks that contain no edges. When such is detected, the block is classified according to its quantized mean. The classifier may be implemented conveniently by first subtracting the mean from each input block and then computing the inner product of that block with edge-oriented template blocks composed of "+1's" and "−1's". The template producing the largest inner product determines the class of the input block. If this inner product is below a pre-determined threshold, the block is declared a non-edge or shade block. The mean classifier is then used to subdivide shade blocks into shade classes based on the mean.

Each state in the FSVQ has a codebook associated with it that must be designed. This can be done in several ways using conventional VQ codebook design methods like the LBG. To start, training images are used to extract training data. The training set is then partitioned according to the true classes of the original input blocks, and codebooks are designed for each of the states. This approach is similar to the omniscient FSVQ design algorithm used in [12, 10, 18] in the sense that the state is obtained from the true classes of the adjacent blocks and not the classes of the coded adjacent blocks. For practical operation, the encoder and decoder will use coded blocks for state determination. Some improvement may be achieved if this fact is reflected in the design procedure [10].

5 Finite-State RVQ

Since FSVQ is able to exploit inter-block correlation among the upper and left most vector blocks, its performance tends to be significantly better than that of conventional VQ. The drawback is that the memory requirement for codebook storage is also significantly greater. For example, consider an FSVQ system with 10 classes, $4 \times 4$ vector size, and 0.5 bit/pixel rate. Each state codebook contains 256 16-dimensional
vectors. The 10 classes imply 100 state codebooks. Therefore 25,600 vectors or equivalently 409,600 pixels must be stored in both the encoder and decoder. This is two-orders of magnitude more storage than that required for conventional VQ.

Here we introduce a new vector quantizer called “finite-state RVQ” or FS-RVQ which is a merger of the finite-state concept and residual VQ. The motivation is that such a merger allows performance comparable to FSVQ to be achieved but with the dramatic memory savings associated with RVQ.

FS-RVQ is very similar to FSVQ except that the unconstrained VQ state codebooks are replaced with RVQ codebooks. All the FSVQ techniques described above can be used to design FS-RVQ systems. Specifically, a large training set is partitioned according to the classes of the original input blocks, and state codebooks are designed using the $M$-search RVQ design algorithm.

The very low memory demands of RVQ make potentially more efficient large-vector-based schemes practical. However, there are two notable issues inherent in such schemes. First, while rate-distortion theory predicts improved performance gain with increasing vector (block) size, inter-block correlation in an image decreases with increasing block size. In other words, the gain due directly to finite-state prediction is inversely related to vector size [29]. Second, large blocks in an image often contain more detail than can be modeled by a single edge as assumed. Thus the adequacy of the classifier becomes an issue and is apt to take on greater complexity in order to more fully exploit the remaining inter-block correlations. Hence, there exists some best choice for vector size with respect to the gains achievable using finite-state prediction and large vector sizes.

If the encoder is permitted to search a small number of state codebooks (more than one) and explicitly send the best state via a separate side information channel, the FS-RVQ system becomes a “forward adaptive” system. This technique has the effect of both finding a better match for an input vector and improving the state transition (leading to a better state sequence). Of course, this comes at the expense of an increase in bit rate. But the amount of increase can be made very small by
using relatively large vector sizes. As we shall see, the additional gain in performance is generally very significant.

6 RVQ and FS-RVQ with Variable Rate

In addition to finite state prediction there is another feature of RVQ that can be used to achieve additional performance improvement. RVQ has progressive coding capability. That is, each stage contributes bits that further improve the reproduction quality. The spatially variant nature of the blocks within an image suggests that a significant improvement in performance can be achieved by incorporating the variable rate capability into the design of FS-RVQ systems.

There are several ways of incorporating the variable rate capability into a finite state system. Here we focus primarily on two variations of variable rate FS-RVQ. The first is where the state codebooks have different sizes with the size determined by the statistical properties of the source. This results in a variable rate system where the rate variation is a function of the state. This approach was considered previously in [21] in the context of conventional FSVQ. With FS-RVQ, however, there are some additional advantages in the design process and in terms of performance via larger vector dimensions.

The second variation of variable rate FS-RVQ is based on making the RVQ rate variant independent of the state specification. In other words, each vector is coded at the lowest rate such that a specified level of quality is maintained. This is shown to be a better approach and is unique to RVQ.

To begin, we consider the first of these approaches where the finite state codebooks are designed with different sizes. Given that there are different rates for the state codebooks, the bit rate of the overall system is given by

\[ r = \sum_{s=1}^{K} p_s r_s, \]  

where \( r_s \) is the bit rate of the RVQ associated with a state \( s \), \( p_s \) is the probability of occurrence of state \( s \), and \( K \) is the number of states.
The design of this type of variable rate FS-RVQ is complicated due, in part, to the large number of design parameters. The vector size, the stage codebook size, the number of stages, the number of paths used in the M-search, the number of states, and the state bit rates are all variables. These variables affect, in a dependent way, the implementation complexity and the image reproduction quality of the system. Nonetheless, it is not particularly difficult to design systems that perform well.

The number of bits allocated to each of the $K$ states can be determined by minimizing the average distortion

$$D = \sum_{s=1}^{K} p_s D_{s,r_s}$$

subject to the constraint that

$$\sum_{s=1}^{K} p_s r_s \leq B_{avg}$$

where $r_s$ is the number of bits required to achieve a mean square distortion $D_{s,r_s}$ for the state $s$, and $B_{avg}$ is the maximum allowed average bit rate (in bits per vector). Note that this bit rate assignment problem is a generalization of the classical bit allocation problem treated in [39].

One approach to finding an optimal or near-optimal bit allocation is to determine the minimum of equation (23) by exhaustively searching all possibilities. In practice, however, the number of possible state assignments may be astronomically large. Thus a more efficient solution is needed. Indeed, several efficient algorithms have been developed [41, 37].

The algorithm derived in [37] is based on a general technique that also leads to the design of unbalanced tree codebooks. As reported in [21], this algorithm can be used to design variable rate finite state VQ systems. In the procedure, a tree is constructed where the root node has $K$ children, one per state, and the subtree rooted at each child $s$ is a unary tree of length $B_s$. Thus the tree has $K$ branches where each branch is associated with a state $s$. Similarly, each branch has $B_s$ nodes where each node $(s,p)$ corresponds to a rate-distortion pair $(b_{s,p}, D_{s,b_{s,p}})$ where ($\forall s \in S$)

$$b_{s,1} < b_{s,2} < \ldots < b_{s,B_s}$$
and
\[ D_{s,b_s,1} \geq D_{s,b_s,2} \geq \ldots \geq D_{s,b_s,B_s} \] (26)

Therefore, the node \((s, 1)\) (the node closest to the root node) corresponds to the pair \((b_{s,1}, D_{s,b_s,1})\) and hence has a bit rate of \(b_{s,1}\) bits per vector, and the node \((s, B_s)\) (the node farthest from the root node) corresponds to the pair \((b_{s,B_s}, D_{s,b_s,B_s})\) and hence has a bit rate of \(b_{s,B_s}\) bits per vector. A four-state bit allocation tree is depicted in Figure 10.

Let \(R\) be a pruned subtree of the constructed tree \(T\), where the branch associated with state \(s\) has now a length \(l_s\). A variable rate FS-RVQ can be easily constructed from the pruned tree \(R\). For each \(s \in S\), the length \(l_s\) denotes the average number of bits used to represent the input vector in state \(s\), where

\[ b_{s,1} \leq l_s \leq b_{s,B_s} \] (27)

The average rate of the variable rate FS-RVQ associated with \(R\) is

\[ r = l(R) = \sum_{s=1}^{K} p_s l_s \] (28)

and the average distortion is

\[ D = \delta(R) = \sum_{s=1}^{K} p_s D_{s,l_s} \] (29)

The bit rate assignment problem can now be solved by finding the rates \(l_1^*, l_2^*, \ldots, l_s^*\) that minimize \(\delta(R)\) subject to \(l(R) \leq B_{\text{avg}}\) over all pruned subtrees \(R \preceq T\). The optimal pruning algorithm described in [37] gives such bit rates.

If the state quantizers are unconstrained VQ’s, several codebooks have to be designed in order to determine the rate-distortion pairs \((b_{s,1}, D_{s,b_s,2}), \ldots, (b_{s,1}, D_{s,b_s,B_s})\). These codebooks can be designed by the splitting LBG algorithm [32]. However, the design complexity is often very large. In the FS-RVQ system, the state quantizers are RVQ codebooks, and the multistage structure is exploited so that the rate-distortion pairs are determined during the design process. Since every RVQ stage represents a
small fraction of the total bit rate, all rate-distortion pairs are generated at the end of the RVQ design process. Figure 11 shows the same four-state bit allocation tree where the nodes of the unary tree are replaced by RVQ stages. Since BRVQ needs only 1 bit/stage, it produces the largest number of rate-distortion pairs that can be determined during the design process.

In the above scheme, a different bit rate (in general) is allocated for each state. However, the bit rate is fixed within each state. This is unnecessarily restrictive in light of the inherent progressive rate capability of RVQ. For a given state, the encoding rate can be made spatially variable resulting in variable rate RVQ state codebooks. For each state, the $M$-search variable rate RVQ design algorithm can be used to design a variable rate codebook with an average distortion $D_{a,t}$ and a peak bit rate $B_s$. The bit rate then varies from $b_{a,1}$ to $B_s$ and takes a very large number of values because the rates are not constrained to be integers. Although side information must be sent to the decoder, this scheme can be effective when relatively large vector sizes (e.g. $8 \times 8$) are used.

As a side note we mention that another approach to constructing a variable rate RVQ for each state is simply to combine a fixed rate state RVQ with a variable rate lossless coder (such as a Huffman coder). This can be done by considering the RVQ equivalent code vectors to be symbols in an extended source alphabet and constructing a variable length noiseless code for these symbols. The bit rates of each state are then replaced by average entropies. As noted earlier, the output entropy of the RVQ is significantly lower than that of the unconstrained ESVQ, and a bit rate much smaller than the logarithm of the number of RVQ equivalent code vectors may be achieved. When small vector sizes (e.g. $4 \times 4$) are used, significant gains can be obtained and may well offset the additional complexity associated with noiseless coding.
7 Experiments and Results

FS-RVQ systems were designed and tested for coding 8-bit monochrome images of size 512 × 512. In all FS-RVQ experiments, 12 images taken from the USC database were used to generate the training set. The test image Lena, shown in Figure 12, was not included in the training set. Since there are usually a large number of states, the training set has to be very large. While the number of natural images available in the database is relatively small, additional vectors may be obtained by rotating the training vectors in the image plane.

Practical FSVQ systems discussed previously in the literature were limited to relatively small vector sizes (typically 4 × 4). However, because of the low memory demands of RVQ, large vector sizes can be used in the design of FS-RVQ. As noted earlier, the RVQ performance increases with increasing vector size. On the other hand, the coding gain due to finite-state prediction decreases when large vector sizes are used. In this work, FS-RVQ systems based on two different vector sizes, (4 × 4) and (8 × 8), are studied.

The state machine used in these experiments employs a classifier which classifies both the vector above and the vector to the left of the current vector into one of 11 classes (three shade classes and eight edge classes). This results in $11^2 = 121$ states.

Systems based on 4 × 4 vectors were first investigated where the SS-RVQ design algorithm was used to design the state codebooks. Each state codebook contained 3 stage codebooks of size 16, leading to an encoding rate of 0.75 bpp. The average entropy of all state codebooks is 0.4509 bpp. Figure 13 shows the coded image Lena at the fixed bit rate of 0.75 bpp, where subjective quality is good. While conventional FSVQ achieves better quality for the same bit rate, memory requirements as well as encoding complexity are substantially greater. For the same bit rate and number of states, FSVQ needs to store about 500,000 vectors and requires 1024 vector distortion calculations per input vector for codebook search. If tree-search VQ is used to offset the computational complexity, the memory required is essentially doubled.
In contrast, FS-RVQ requires only 5808 vectors be stored and 48 vector distortion calculations for the search. The fixed rate FS-RVQ designed above was also combined with a Huffman coder, leading to a modest increase in complexity. However, the bit rate for coding Lena was reduced to 0.4588 bpp.

In the second set of experiments, 8 x 8 vectors were used in the design of FS-RVQ. The $M$-search RVQ design algorithm with $M = 4$ was used to design the residual state codebooks. These codebooks contained 8 stage codebooks each containing 4 code vectors, leading to an encoding rate of 0.25 bpp. Figures 14 and 15 show the coded image Lena using memoryless RVQ and FS-RVQ, respectively. The performance of FS-RVQ both in terms of PSNR and subjective quality is seen to be much better than that of memoryless RVQ, both of which are better than conventional VQ with 4 x 4 vectors.

In the last set of experiments, the variable rate $M$-search FS-RVQ design algorithm is used to design variable size RVQ state codebooks with 4 code vectors per stage and $M = 4$. The maximum bit rate (in bits per vector) $B_s$ is determined based on the size of the training set associated with the state $s$. For each state $s$, the size of the training set is a reasonable estimate of the probability of occurrence of state $s$. The bit rate $B_s$ is determined such that each stage partition cell is richly populated. First, the bit rate was fixed within each state. As expected, the variable size state codebooks produced better reconstruction image quality (shown in Figure 16), both subjectively and objectively. Then, the VR-RVQ design algorithm was used so that the bit rate is now spatially variable. Figure 17 illustrates the additional improvement that can be gained by employing the VR-RVQ design algorithm. Since 8 x 8 vectors are used in the variable rate FS-RVQ design, the additional bit rate needed to send side information is very small, and the overall bit rate is not significantly elevated. In this set of experiments, forward adaptation is used to improve the performance of variable rate FS-RVQ.
8 Conclusions

The design techniques presented have been shown to yield FS-RVQ's that provide good performance when applied to image coding. The good performance of FS-RVQ's designed using $8 \times 8$ vectors indicates that finite-state machines operating on such large vectors can still take advantage of the correlation over a large area in the image plane.

Variable rate coding is achieved here by directly exploiting the FS-RVQ structure. Firstly, each state codebook can have a different size. Side information need not be sent because the state codebooks are available at both the encoder and decoder. Secondly, the multistage RVQ structure can be very easily exploited to produce a variable rate RVQ codebook for each state. Although side information must be sent to the decoder, the additional bit rate is usually not significant.

The results reported in this paper suggest that variable rate FS-RVQ with variable rate state codebooks has the potential of achieving the best performance within this family of VQ. For the popular image Lena, a PSNR of 31.89 dB is achieved at 0.2724 bpp and the reconstructed image quality is rather good subjectively. The superior performance obtained with variable rate FS-RVQ is due to the design philosophy that concentrates the coding power in the regions where it is most needed.

The performance of FS-RVQ can be significantly improved by allowing the encoder to search a small number of state codebooks before making its decision. This implies sending side information, which can be very costly when small vectors are used. On the other hand, the additional bit rate is not significant when large vectors (e.g. $8 \times 8$) are used. This scheme combines both backward adaptation (due to finite-state prediction) and forward adaptation (with side information), and does not require the long delays of most trellis encoders.
References


27


Figure 1: A block diagram of a vector quantizer

Figure 2: A two-stage residual vector quantizer
Figure 3: Binary residual VQ (BRVQ) structure

Figure 4: BRVQ tree structure
Table 1: Memory, encoder complexity, decoder complexity and performance of ESVQ, TSVQ, ES-RVQ, SS-RVQ and MS-RVQ

<table>
<thead>
<tr>
<th></th>
<th>ESVQ</th>
<th>TSVQ</th>
<th>SS-RVQ</th>
<th>ES-RVQ</th>
<th>MS-RVQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>large</td>
<td>larger</td>
<td>very small</td>
<td>very small</td>
<td>very small</td>
</tr>
<tr>
<td>Encoder Complexity</td>
<td>large</td>
<td>small</td>
<td>small</td>
<td>large</td>
<td>moderate</td>
</tr>
<tr>
<td>Decoder Complexity</td>
<td>very small</td>
<td>very small</td>
<td>small</td>
<td>small</td>
<td>small</td>
</tr>
<tr>
<td>Performance</td>
<td>very good</td>
<td>very good</td>
<td>moderate</td>
<td>good</td>
<td>good</td>
</tr>
</tbody>
</table>
Figure 5: PSNR vs N where N is the number of vectors per stage. The bit rate is fixed at 0.25 bpp. The vector size is fixed at $16 \times 16$. 

\[ \text{Peak signal-to-noise ratio in decibels} \]

\[ \text{Number of vectors per stage} \]
Figure 6: PSNR performance of two codebooks, one designed using Juang and Gray's method (Bottom), the other using the sequential search RVQ method (Top), for the test image Lena when $4 \times 4$ (Figure 6a) and $8 \times 8$ (Figure 6b) vectors are used.
Figure 7: PSNR performance of two codebooks, one designed using Juang and Gray's method (Bottom), the other using the MS-RVQ ($M = 8$) method (Top), for the test image Lena. The vector size is fixed at $4 \times 4$. 
<table>
<thead>
<tr>
<th></th>
<th>PSNR(db)</th>
<th>Cbk entropy(bpp)</th>
<th>Cbk storage(vecs)</th>
<th>Codevecs/search</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESVQ</td>
<td>31.81</td>
<td>0.5148</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td>RVQ(2,1)</td>
<td>28.19</td>
<td>0.2720</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>RVQ(2,2)</td>
<td>28.78</td>
<td>0.2859</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>RVQ(2,4)</td>
<td>29.11</td>
<td>0.3123</td>
<td>20</td>
<td>80</td>
</tr>
<tr>
<td>RVQ(2,8)</td>
<td>29.76</td>
<td>0.3599</td>
<td>20</td>
<td>160</td>
</tr>
<tr>
<td>RVQ(4,1)</td>
<td>29.88</td>
<td>0.3502</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>RVQ(4,2)</td>
<td>30.03</td>
<td>0.3519</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>RVQ(4,4)</td>
<td>30.13</td>
<td>0.3673</td>
<td>20</td>
<td>80</td>
</tr>
<tr>
<td>RVQ(4,8)</td>
<td>30.16</td>
<td>0.3677</td>
<td>20</td>
<td>160</td>
</tr>
<tr>
<td>RVQ(32,1)</td>
<td>30.78</td>
<td>0.4508</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>RVQ(32,2)</td>
<td>30.97</td>
<td>0.4564</td>
<td>64</td>
<td>128</td>
</tr>
<tr>
<td>RVQ(32,8)</td>
<td>31.08</td>
<td>0.4573</td>
<td>64</td>
<td>512</td>
</tr>
</tbody>
</table>

Table 2: PSNR performance, entropy, codebook storage and number of search operations of ESVQ and RVQ(N,M) (N is the stage codebook size, M is the number of searching paths) for the image Lena at 0.625 bpp when $4 \times 4$ vectors are used.
<table>
<thead>
<tr>
<th></th>
<th>Bit Rate(bpp)</th>
<th>PSNR(db)</th>
<th>Entropy(bpp)</th>
<th>Memory(bytes)</th>
<th>vecs/search</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESVQ</td>
<td>0.5000</td>
<td>30.17</td>
<td>0.3757</td>
<td>18432</td>
<td>256</td>
</tr>
<tr>
<td>RVQ(2,1)</td>
<td>0.5000</td>
<td>28.02</td>
<td>0.2644</td>
<td>3072</td>
<td>16</td>
</tr>
<tr>
<td>RVQ(2,1)</td>
<td>0.6250</td>
<td>28.29</td>
<td>0.2720</td>
<td>9472</td>
<td>20</td>
</tr>
<tr>
<td>RVQ(2,2)</td>
<td>0.6250</td>
<td>28.98</td>
<td>0.2859</td>
<td>9472</td>
<td>40</td>
</tr>
<tr>
<td>RVQ(4,1)</td>
<td>0.6250</td>
<td>29.88</td>
<td>0.3502</td>
<td>9472</td>
<td>20</td>
</tr>
<tr>
<td>RVQ(4,4)</td>
<td>0.6250</td>
<td>30.13</td>
<td>0.3673</td>
<td>9472</td>
<td>80</td>
</tr>
<tr>
<td>RVQ(4,1)</td>
<td>0.7500</td>
<td>30.64</td>
<td>0.3721</td>
<td>17920</td>
<td>24</td>
</tr>
</tbody>
</table>

Table 3: Bit rate, PSNR performance, entropy, memory (for storing codebook and huffman table) and number of search operations of ESVQ and RVQ (N,M) (N is the stage codebook size, M is the number of searching paths) for the image Lena when 4 × 4 vectors are used.
Figure 8: State specification for vectors

Figure 9: Other more general state specifications
Figure 10: A four-state bit allocation tree
Figure 11: A four-state bit allocation tree. Each branch represents a state. Each level represents a stage in the RVQ.
Figure 12: Original Lena image
See last page for better quality
Figure 13: Lena image coded at 0.75 bpp (fixed rate) and 0.4588 bpp (with Huffman coding), (Sequential search FS-RVQ, 4x4, 4 vectors per stage). PSNR is 33.09 dB. See last page for better quality.
Figure 14: Lena image coded at 0.25 bpp (MS-RVQ, 8x8, 4 vectors per stage, M = 4). PSNR is 28.26 dB. See last page for better quality.
Figure 15: Lena image coded at 0.25 bpp (M-search FS-RVQ, 8x8, 4 vectors per stage, M = 4). PSNR is 29.33 dB. See Last Page for better quality.
Figure 16: Lena image coded at 0.2412 bpp (Variable rate M-search FS-RVQ with forward adaptation, 8x8, 4 vectors per stage, M = 4). Bit rate varies from state to another but fixed within each state. PSNR is 31.34 dB. See Last Page for better quality.
Figure 17: Lena image coded at 0.2724 bpp (Variable rate M-search FS-RVQ with forward adaptation, 8x8, 4 vectors per stage, M = 4). VR-RVQ design algorithm is used, and state bit rate is variable with time. PSNR is 31.89 dB. See last page for better quality.
Image Coding Using Entropy-Constrained Residual Vector Quantization*

Faouzi Kossentini and Mark J. T. Smith
Digital Signal Processing Laboratory
School of Electrical Engineering
Georgia Institute of Technology
Atlanta, GA 30332

Christopher F. Barnes
Georgia Tech Research Institute
Georgia Institute of Technology
Atlanta, GA 30332

April 15, 1993

Abstract

In this paper, the residual vector quantization (RVQ) structure is exploited to produce a variable length codeword RVQ. Necessary conditions for the optimality of this RVQ are presented, and a new entropy-constrained RVQ (EC-RVQ) design algorithm is shown to be very effective in designing RVQ codebooks over a wide range of bit rates and vector sizes. The new EC-RVQ has several important advantages. It can outperform entropy-constrained VQ (EC-VQ) in terms of peak signal-to-noise ratio (PSNR), memory, and computation requirements. It can also be used to design high rate codebooks and codebooks with relatively large vector sizes. Experimental results indicate that when the new EC-RVQ is applied to image coding, very high quality is achieved at relatively low bit rates.

*This work was supported in part by the National Science Foundation under contract MIP-9116113 and by NASA.
1 Introduction

Vector quantization (VQ) has received much attention and is a powerful and effective technique for image compression [9]. A motivation for this approach is that the performance of vector quantizers can approach the distortion-rate bound $D(R)$ as the vector size becomes sufficiently large [3]. However, the rate at which the performance of VQ approaches the bound $D(R)$ as a function of increasing vector size is rather slow [3]. Moreover, both the computation and memory requirements associated with VQ increase exponentially as the vector size increases. Therefore, relatively small vectors, typically of size $4 \times 4$, are usually used in the design of unconstrained exhaustive search VQ codebooks for image coding.

Reducing the large complexity and memory requirements of VQ has been the focus of much research. Various imposed structural constraints have been considered, but such constraints generally lead to reduced performance for a given rate and dimension. However, the reduction in complexity obtained is often a good trade for the moderate loss in quality. Some examples of structured vector quantizers are lattice VQ [7], hierarchical VQ [27], and tree-searched VQ (TSVQ) [4, 10]. Residual vector quantization (RVQ) or multistage VQ is one such structured vector quantizer whose structure reduces both the memory and computation costs, and is able to operate over a large range of bit rates and vector sizes. The recent interest in RVQ is due largely to its good complexity/performance tradeoffs, and to the recent advances made in design methodology, which have resulted in noticeable improvements over previous design methods [14, 26].

The structural constraints of RVQ result in a performance degradation compared to an unconstrained VQ with the same bit rate and vector size. This degradation can be attributed to two factors. First, the RVQ decoder is constrained by a direct-sum codebook structure where all possible output vectors of the RVQ are formed by
the sum of stage code vectors—this set is called the direct-sum codebook. Second, the encoder typically employs an efficient sequential stage-wise search procedure for practical reasons. However, entanglements in the RVQ tree tend to reduce encoding accuracy when fast searching is performed. This difficulty is obviated by exhaustive searching or other forms of optimal sequential searching (see [1]) but the price paid in computational complexity is generally enormous.

Looking beyond this, however, the structure of RVQ has properties that make it attractive. The multi-stage structure can be exploited to produce variable number-of-stages RVQ (one form of variable rate RVQ), which was shown in [19, 20] to lead to improvements in performance over fixed rate RVQ. In addition, the direct-sum structural constraint usually leads to an RVQ output entropy which is much smaller than the logarithm of the number of direct-sum code vectors. Experimental evidence suggests that the decrease in output entropy compensates for the increase in average distortion which, in turn, leads to a very competitive coding system.

A simple approach to constructing another form of variable rate RVQ is to combine a fixed rate RVQ with a noiseless coder. However, a better approach is to directly incorporate entropy coding in the design process. The joint optimization of a VQ and an entropy coder was shown to lead to a significant improvement in performance for the conventional VQ case [5, 6]. This motivates the investigation of an RVQ design algorithm that minimizes the average distortion subject to a constraint on the output entropy of the RVQ. This paper introduces a new entropy-constrained RVQ (EC-RVQ) design algorithm that is very effective in designing variable rate RVQ codebooks. EC-RVQ is shown to be capable of outperforming conventional EC-VQ in terms of computational complexity, memory requirements and coding quality, and has the ability to operate over a much wider range of bit rates and vector sizes.

To set the mathematical notation and terminology used throughout this paper, the next section begins with a brief summary of fixed rate RVQ. To lay the founda-
tion for the discussion of variable rate RVQ and the development of the new EC-RVQ
design algorithm, necessary conditions for the optimality of fixed rate RVQ and cor­
responding design algorithms are also discussed in the section as well. Next, methods
of constructing three forms of variable rate RVQs are discussed and compared in Sec­
tion 3. Necessary conditions for the optimality of variable rate RVQ are presented,
and a discussion of the new EC-RVQ algorithm is considered in Section 4. Section 5
discusses the performance of EC-RVQ when used in image coding applications. The
paper concludes with some general comments on improving EC-RVQ performance
that reflect work presently under study.

2 Fixed Rate RVQ

Residual vector quantization (RVQ) or multistage VQ consists of a cascade of VQ
stages, each operating on the “residual” of the previous stage. A block diagram of a P­
stage RVQ is given in Figure 1 for illustration. A general RVQ consisting of P stages
(with $N_i$ vectors in the ith stage) is capable of uniquely representing $N = \prod_{i=1}^{P} N_i$
vectors with only $\sum_{i=1}^{P} N_i$ code vectors required for storage. Thus, the RVQ achieves
tremendous savings over unconstrained VQ in terms of memory requirements, and
may also achieve similar savings in computations.

To establish the notation and review the key points for optimal fixed rate RVQ,
let $\mathbf{z}_1$ be a realization of the random $k$-dimensional vector $\mathbf{X}_1$ described by the prob­
ability density function (pdf) $f_{\mathbf{X}_1}(\mathbf{z}_1)$ on $\mathbb{R}^k$ and assume this to be the input to the
$P$-stage RVQ shown in Figure 1. For the $p$th stage VQ with $1 \leq p \leq P$, let us define
the following symbols:
Figure 1: A $P$-stage residual vector quantizer

$N_p$ the $p$th stage codebook size (number of codebook vectors)

$j_p$ the $p$th stage index: $\{0 \leq j_p \leq N_p - 1\}$

$J_p$ the $p$th set of all possible values for $j_p$: i.e. $\{0, 1, 2, \ldots, N_p - 1\}$

$y_p(j_p)$ the $j_p$th code vector

$S_p(j_p)$ the $j_p$th partition cell

$V_p(j_p)$ the $j_p$th conditional-stage residual cell

$C_p$ the $p$th stage codebook $\{y_p(j_p) : j_p \in J_p\}$

$P_p$ the $p$th stage partition $\{S_p(j_p) : j_p \in J_p\}$

$Q_p$ the $p$th stage quantizer mapping

Associated with a $P$-stage RVQ is an equivalent single-stage direct-sum VQ. The direct-sum VQ and RVQ are identical in the sense that they produce the same representation of the source output and they have the same expected distortion. For the direct-sum VQ, let us define the following symbols:

$N$ direct-sum codebook size ($N = \prod_{i=1}^{P} N_i$)

$J$ direct-sum $P$-tuple index set $J_1 \times J_2 \times \cdots \times J_P$

$j$ a $P$-tuple index in $J$

$y(j)$ $j$th direct-sum code vector

$V(j)$ $j$th direct-sum partition cell

$C$ direct-sum codebook $\{y(j) : j \in J\}$

$P$ direct-sum partition $\{V(j) : j \in J\}$

$Q$ direct-sum mapping $Q(x_1) = \sum_{p=1}^{P} Q_p(x_p)$
The direct-sum codebook contains all possible ordered sums of the stage code vectors, i.e., \( C = C_1 \oplus C_2 \oplus \ldots \oplus C_P \). The direct-sum code vectors are given by \( y(j) = \sum_{p=1}^{P} y_p(j_p) \), where \( j_p \) is the \( p \)th member of the ordered \( P \)-tuple index \( j \).

The direct-sum VQ quantizes the source vector \( x_1 \) and outputs the representation \( \hat{x}_1 = Q(x_1) \) given by \( Q(x_1) = \sum_{p=1}^{P} Q_p(x_p) \), where we call \( x_p = x_1 - \sum_{i=1}^{p-1} Q_i(x_i) \) the \( p \)th stage causal residual. The term causal refers to the stages supporting the computation of the residual; i.e., the stage residuals are computed sequentially starting from the first stage to the \( p \)th stage.

To formalize this notion, let the distortion that results from representing the input \( x_1 \) by the quantized output \( \hat{x}_1 \) be expressed by \( d(x_1, \hat{x}_1) \). The distortion measure \( d(x, y) \) is assumed to be a non-negative real-valued function that satisfies the following requirements:

1. For any fixed \( x \in \mathbb{R}^k \), \( d(x, y) \) is a continuously differentiable function of \( y \in \mathbb{R}^k \).
2. \( d(x, y) \) is translationally invariant.
3. For any fixed \( x \in \mathbb{R}^k \), \( d(x, y) \) is a strictly convex function of \( y \), that is, \( \forall y_1, y_2 \in \mathbb{R}^k \) and \( \lambda \in (0, 1) \), \( d(x, \lambda y_1 + (1 - \lambda)y_2) < \lambda d(x, y_1) + (1 - \lambda)d(x, y_2) \).

A \( P \)-stage RVQ is said to be optimal if it gives at least a locally minimum value of the average distortion. There are two necessary conditions for the optimality of fixed rate RVQ [1, 2]. First, the encoder must map the input vectors according to the following nearest-neighbor rule:

\[
x_1 \in V^*(j) \quad \text{if and only if} \quad d(x_1, y(j)) \leq d(x_1, y(k)) \quad \text{for all} \quad k \in J.
\]  

Second, the stage code vectors \( y_p(j_p) \) at the \( p \)th stage must satisfy [2, 23]

\[
\int d\left(\gamma_p, y^*_p(j_p)\right) f_{\Gamma_p\mid j_p}(\gamma_p)d\gamma_p = \inf_{u \in \mathbb{R}^k} \int d(\gamma_p, u)f_{\Gamma_p\mid j_p}(\gamma_p)d\gamma_p < \infty
\]
where $\gamma_p = x_1 - \sum_{i=1}^{p} y_i(j_i)$ is a realization of the conditional-stage residual random vector $\Gamma_p$, and the pdf $f_{\Gamma_p}(\gamma_p)$ is related to the source pdf $f_{X_1}(\cdot)$ according to

$$f_{\Gamma_p}(\gamma_p) = \frac{\sum_{j \in H_p(j_p)} I(V(j)) f_{X_1}(g(\beta_p(j)) + \gamma_p)}{pr(\gamma_p \in V_p(j_p))},$$

(3)

where $\beta_p(j) = (j_1, j_2, \ldots, j_{p-1}, j_p+1, \ldots, j_p)$, $g(\beta_p(j)) = \sum_{i=1}^{p} y_i(j_i)$, $H_p(j_p) \subset J$ is the set of all indices $j = (k_1, k_2, \ldots, k_{p-1}, j_p, k_{p+1}, \ldots, k_p)$ such that $j_p \in J_p$, and $I[V(j)]$ is an indicator function for the direct-sum partition cell $V(j)$, that is, $I[V(j)] = 1$ if $x_1 \in V(j)$ and $I[V(j)] = 0$ otherwise. The $y_p(j_p)$'s which satisfy equation (2) are generalized centroids of conditional-stage residual vectors (i.e., residual vectors formed from the encodings of all prior and subsequent RVQ stages). Hence, the second condition will be referred to as the conditional-stage residual centroid condition hereafter. A mathematical derivation of these two conditions is given in [2, 23].

### 2.1 The Fixed Rate RVQ Design Algorithm

The fixed rate RVQ design algorithm, introduced in [1], attempts to optimize all stage codebooks jointly to minimize the reconstruction error over all training data subject to a constraint on the number of direct-sum code vectors. Assuming that all stage codebooks are held fixed, optimization of the encoder implies that each training set vector is mapped to its closest direct-sum code vector using the nearest-neighbor rule (1). In general, this can be accomplished by exhaustively searching the direct sum codebook. However, this technique typically carries sufficient computational overhead to be unattractive. An alternative approach is to sequentially search the RVQ stage codebooks. This technique results in an increase in speed, but unfortunately leads to a significant degradation in performance since optimal code vector selection in the direct-sum codebook is no longer guaranteed. To address this issue, the $M$-search technique was explored and was shown to be very efficient when used to search the
RVQ tree [1, 18]. Small improvements can be obtained by simply using $M$-search when encoding the input using a sequentially-designed RVQ codebook. However, better results can be obtained by directly incorporating the $M$-search in the RVQ design as well as in the encoder [2, 18]. An additional gain can be achieved for the same complexity by allowing the value of $M$ to be larger in some stages of the RVQ and smaller in others. This can be done by first defining a desired level for the average number of $M$-search computations. Using a large training set, the best value of $M$ for each stage can be determined empirically such that the total number of $M$-search computations is within the pre-specified tolerance.

Given a fixed direct-sum partition, the fixed rate RVQ design method used in this work is simply an iterative Gauss-Seidel algorithm that jointly optimizes the stage codebooks by successively operating on each RVQ stage while holding fixed all other stage codebooks. At each stage optimization step, code vectors are found that simultaneously satisfy the conditional-stage residual centroid condition (2). Assuming that the squared error distortion measure is used, each "decoder-only" iteration will update the stage codebooks such that the average distortion will either be reduced or left unchanged [24]. Using theorems in [11], it can be shown [24] that if the encoder yields a Voronoi partition with respect to the direct-sum codebook, then the fixed rate RVQ design algorithm converges monotonically to a fixed point which satisfies the necessary conditions (1) and (2) for minimum squared error distortion.

This proven convergence behavior is based on an exhaustive search encoder, which is not realistic for a practical system in general. For practical applications, a sequential nearest neighbor or an $M$-search encoder is used. In these cases, the encoder optimization step may actually increase the average distortion and monotonic convergence cannot be guaranteed. However, experimental results have shown that the sequential-search RVQ design algorithm effectively reduces the average distortion with only occasional deviations from monotonicity. Furthermore, in all our experiments,
the $M$-search RVQ design algorithm converged monotonically to a local minimum, even when relatively small values of $M$ (such as 2 or 3) were used.

2.2 Comments on RVQ Performance

An upper bound on the performance of fixed rate RVQ is the performance of exhaustive-search VQ [26]. For the same bit rate and vector size (i.e., same number of code vectors), the average distortion introduced by the RVQ can be shown to be generally larger than that introduced by an unconstrained VQ. For example, let’s assume that a conventional VQ and an RVQ have the same fixed partition of $\mathbb{R}^k$. It is shown in [11] that the average distortion can be minimized if and only if the code vectors are selected as the centroids of their respective partition cells. Since the code vectors in the conventional VQ codebook are structurally independent, this selection can be done separately for each partition cell. However, code vectors formed by direct-sums of stage code vectors are structurally dependent and hence it is unlikely all will be centroids of their respective direct-sum partition cells. As a matter of fact, these direct-sum code vectors are not guaranteed to even lie within their respective cells. Therefore, the average distortion of RVQ is higher than that of conventional VQ.

However, by using large vector sizes and multi-path searching, the RVQ performance is shown to exceed that of conventional VQ with only a fraction of the computation and memory requirements [18]. Moreover, the direct-sum codebook constraint usually leads to an output entropy $H$ that is smaller than that of unconstrained VQ. This can be easily demonstrated using the fact that the joint entropy of a collection of sources (or random variables) is less than or equal to the sum of the entropies of the individual sources [8]. That is, given $P$ random variables $X_1, \ldots, X_P$,

$$H(X_1, \ldots, X_P) \leq \sum_{p=1}^{P} H(X_p).$$

Given the set of $P$-tuple indices $\mathbf{J}$, one can uniquely index all the code vectors in an unconstrained VQ codebook (which has the same number of code vectors as the
direct-sum RVQ codebook) by the mapping \( \gamma : J_1 \times \ldots \times J_P \mapsto J \), where

\[
\gamma(j_1, \ldots, j_P) = \sum_{p=1}^{P} j_p \prod_{k=0}^{p-1} |J_k|
\]

where \(|J_0| = 1\) and \(|J_k|\) is the size of the set \(J_k\). Since the \(j_1, \ldots, j_P\) are independent (they are chosen arbitrarily), the output entropy of the unconstrained VQ is

\[
H(J) = \sum_{p=1}^{P} H(J_p),
\]

where \(H(J_p)\) is the entropy of \(J_p\). One can also use the same indexing scheme to index the direct-sum codebook, except that now \(j_p\) denotes the index for the \(p\)th stage of the RVQ. As noted earlier, the RVQ stages are related by the direct-sum structure, and \(j_1, \ldots, j_P\) are not independent. Thus, \(H_{RVQ}(J) = H(J_1, \ldots, J_P) \leq \sum_{p=1}^{P} H(J_p) = H_{VQ}(J)\). Notice that this result can also be obtained by using the fact that the entropy of the collection of the random variables \(J_1, J_2, \ldots, J_P\) is equal to the sum of the conditional entropies, i.e.,

\[
H(J_1, J_2, \ldots, J_P) = \sum_{p=1}^{P} H(J_p|J_{p-1}, \ldots, J_1).
\]

The previous results suggest that the direct-sum codebook constraints can generally be expected to lead to both an increased average distortion and a decreased output entropy. This implies that for a given average bit rate, variable rate RVQ could conceivably have the potential to be competitive with variable rate VQ.

### 3 Variable Rate RVQ

For a given vector size \(k\), variable rate VQ implementations are those that, if properly designed, can operate at bit rates close to the ones given by the \(k\)th order rate-distortion curve \(R_k(D)\) of the input. There are several ways in which a variable rate RVQ can be constructed. As reported in [19], a variable rate implementation can be
achieved by exploiting the inherent multi-stage structure of RVQ. Since each stage contributes independently to the total bit rate, variable rate coding can be achieved easily by truncating the number of RVQ stages used for a given source vector. For each input vector, the encoding terminates once the distortion falls below a prescribed threshold. Clearly, the encoder and the decoder must both have knowledge of the number of stages (bit rate) used to encode a given vector. Sending such a rate to the decoder is usually done by sending side information, which can be very costly. However, when relatively large vector sizes (such as 8 x 8 or 16 x 16) are used, side information requires only a small fraction of the total bit rate [19]. This variable rate technique has two advantages: 1) Incorporating such a technique into the RVQ design algorithm leads to reduced encoding complexity because fewer distortion calculations are needed to encode vectors with low variances, and the centroid computation requires fewer additions; and 2) variable rate RVQ of this type tends to allow for a better match to the statistics of images. A large number of bits can be used to encode edge vectors while a small number can be used to encode low variance vectors [19].

Another approach to variable rate RVQ is to entropy code the RVQ output indices. In this case, a fixed rate RVQ is combined with a variable rate lossless coder (such as a Huffman coder). This can be done by considering the RVQ direct-sum code vectors to be symbols in an extended source alphabet and constructing a variable length lossless code for them. The complicated interdependencies among the stages of an RVQ often results in a direct-sum codebook where the code vectors have a very nonuniform probability distribution. Therefore, the output entropy of RVQ is usually much smaller than the logarithm of the number of direct-sum code vectors. Experimental results, reported in [20], show that the output entropy of the direct-sum codebook is much smaller than that of the unconstrained VQ codebook (for the same number of code vectors). Thus the RVQ/entropy coder combination can lead to a
substantially lower average bit rate while maintaining the same performance level of a fixed rate RVQ.

A superior approach to the variable rate RVQ implementation described above is one in which all code vectors and codewords are optimized with respect to each other. Therefore, the natural design problem for entropy-based RVQ is to find a direct-sum codebook whose vectors minimize the average reconstruction error over all training set data subject to a constraint on the output entropy of the RVQ. In the next section, necessary conditions for the optimality of variable rate RVQ are presented, an entropy constrained RVQ (EC-RVQ) design algorithm which satisfies these conditions is introduced, and the performance of this algorithm is demonstrated and discussed.

4 Entropy-Constrained RVQ

The high level structure of the EC-RVQ is illustrated in Figure 2. It consists of a P-stage RVQ where the stage codewords are input to a mapping operator. The mapping operator transforms the direct-sum index $j = (j_1, j_2, \ldots, j_P)$ codeword into a variable length codeword $c(j)$ that is then used as the representation of the compressed data. The mapping operator can be an entropy coder or a collection of stage entropy coders. The idea underlying the entropy mapping operation is that $j$'s that occur very often are represented with short codewords and $j$'s that occur infrequently are represented with longer codewords such that the average bit rate is reduced.

4.1 Necessary Conditions for Optimal Variable Rate RVQ

For the direct-sum VQ, let $J$ be the set of variable length indices $\{c(j), j \in J\}$. The direct-sum VQ, $Q : \mathbb{R}^k \rightarrow C$, quantizes the source vector $z_1$ and outputs $Q(z_1)$, and may be realized by a composition of a variable length encoder mapping $E : \mathbb{R}^k \rightarrow J$
INPUT $J = \{1, 2, \ldots, J_p\}$

WHERE

$\mathcal{E}(z_1) = \mathbb{J}(j)$ if and only if $z_1 \in \mathbb{V}(j)$,

and a variable length decoder mapping $\mathcal{D} : J \mapsto \mathbb{C}$ where

$$\mathcal{D}(\mathbb{J}(j)) = \mathbb{Y}(j).$$

The variable length encoder can be further decomposed into two mappings, $\mathcal{E} = \mathbb{L} \circ \mathbb{E}$, where $\mathbb{E} : \mathbb{R}^k \mapsto J$ and $\mathbb{L} : J \mapsto \mathbb{J}$, and $\circ$ denotes composition. Similarly, one can decompose the variable length decoder into two mappings, $\mathcal{D} = \mathbb{D} \circ \mathbb{L}^{-1}$, where $\mathbb{L}^{-1} : \mathbb{J} \mapsto J$, and $\mathbb{D} : J \mapsto \mathbb{C}$. Note that the mapping $\mathbb{L}$ is an invertible mapping with inverse $\mathbb{L}^{-1}$.

Let $x_1$ be a realization of the random $k$-dimensional vector $X_1$ described by the probability density function (pdf) $f_{X_1}(x_1)$ on $\mathbb{R}^k$. Also, let the distortion that results from representing $x_1$ with $\hat{x}_1$ be expressed by $d(x_1, \hat{x}_1)$. The distortion measure
\(d(x, y)\) is assumed to be a non-negative real valued function that satisfies the requirements (1)-(3) in Section 2. A variable rate \(P\)-stage RVQ (with an average rate \(\leq R\)) is said to be optimal for \(f_{\mathbf{X}_1}(.)\) if it gives a locally or globally minimum value of the average distortion. The design problem can be stated as follows: Choose the codebook \(C\), partition \(P\) and mapping \(L\) that minimize the Lagrangian

\[
J_\lambda(E, L, D) = E\{d(x_1, x_1) + \lambda |L(j)|\}
\]

where \(\lambda\) is the Lagrange multiplier and \(|L(j)|\) denotes the length of \(L(j)\).

There are three necessary conditions for the optimality of variable rate RVQ [23]. First, the encoder must map the input vectors according to the following nearest-neighbor encoding rule:

\[
x_1 \in V^*(j) \iff d(x_1, y(j)) + \lambda |L(j)| \leq d(x_1, y(k)) + \lambda |L(k)| \text{ for all } k \in J.
\]

Second, the mapping \(L\) must be one that minimizes the expected codeword length, \(R = \sum_{j \in J} |L(j)| \text{pr}(j)\), where \(\text{pr}(j) = \text{pr}(x_1 \in V(j))\). Setting the codeword length \(|L(j)|\) to

\[
|L^*(j)| = -\log_2 \text{pr}(j) = -\log_2 \text{pr}(j_1, j_2, \ldots, j_P)
\]

results in an average rate which is equal to the output entropy of the direct sum RVQ. Third, the stage code vectors \(y_p(j_p)\) at the \(p\)th stage must satisfy the conditional-stage residual centroid condition (2). A complete derivation of these conditions is involved and may be found in [23].

The probability \(\text{pr}(j_1, j_2, \ldots, j_P)\) of a path in the RVQ can also be written as the product of conditional probabilities, i.e.

\[
\text{pr}(j_1, j_2, \ldots, j_P) = \text{pr}(j_P|j_{P-1}, \ldots, j_1) \text{pr}(j_{P-1}|j_{P-2}, \ldots, j_1) \ldots \text{pr}(j_2|j_1) \text{ pr}(j_1)
\]

Therefore,

\[
|L^*(j)| = -\log_2 \text{pr}(j_P|j_{P-1}, \ldots, j_1) - \log_2 \text{pr}(j_{P-1}|j_{P-2}, \ldots, j_1) - \ldots - \log_2 \text{pr}(j_2|j_1) - \log_2 \text{pr}(j_1)
\]
and

\[ H^*(J_1, J_2, \ldots, J_p) = \sum_{p=1}^{P} H(J_p | J_{p-1}, \ldots, J_1). \]

### 4.2 The EC-RVQ Design Algorithm

The EC-RVQ design algorithm is an iterative descent algorithm similar to the one used for the design of EC-VQ codebooks. Each iteration consists of applying the transformation

\[ (E(t+1), L(t+1), D(t+1)) = T(E(t), L(t), D(t)) \]

where

\[ E(t+1) = \arg \min_E (E, L(t), D(t)) \] (optimum partitions)
\[ L(t+1) = \arg \min_L (E(t+1), L, D(t)) \] (optimum codeword lengths)
\[ D(t+1) = \arg \min_D (E(t+1), L(t+1), D) \] (optimum code vectors)

Following the lines of argument of [5], one can show [24] that every limit point of the sequence \((E(t), L(t), D(t)), t = 0, 1, \ldots\), generated by the transformation \(T\) minimizes the Lagrangian \(J_\lambda(E, L, D)\) (as given by (4)). Therefore, the EC-RVQ design algorithm is guaranteed to converge to a local minimum.

To find the entire convex hull of the operational rate-distortion curve, the minimization of \(J_\lambda(E, L, D)\) is repeated for various \(\lambda\)'s. Starting with \(\lambda = 0\) (which corresponds to the RVQ codebook designed by the fixed rate RVQ design algorithm), the EC-RVQ design algorithm uses a pre-determined sequence of \(\lambda\)'s [5] to design variable rate EC-RVQ codebooks. A summary of the algorithm is given in Figure 3.

As in the design of fixed rate RVQ codebooks, multipath searching is used in the encoder optimization step of the EC-RVQ design algorithm to closely satisfy the encoding rule given by (5). The \(M\)-search algorithm is found to be very efficient in substantially reducing the encoding complexity of EC-RVQ for only a small loss in
Figure 3: The EC-RVQ design algorithm
performance. Also, the Gauss-Seidel algorithm is used to find optimal stage code vectors (i.e., stage code vectors that simultaneously satisfy the conditional-stage residual centroid condition (2)).

Unique to EC-RVQ is optimization of the lengths of the codewords which represent direct-sum partition cells or code vectors. Allowing the use of non-integer codeword lengths, the self-information of a $P$-tuple index (or random variable) $j = (j_1, j_2, \ldots, j_P)$, given by (6), is essentially the optimal length of the variable length codeword associated with that index $j$. Equation (7) shows that such an optimal length is also the sum of $P$ stage conditional self-information components. Because of the dependencies that usually exist between the stages of the RVQ, observations of past encoding decisions provides some partial information about the $p$th stage index $j_p$. While the estimation problem is difficult, one can still find a good estimate of the lengths of variable-length stage codewords by using a sufficiently large training set.

It is evident that the aggregate number of tables of conditional-stage entropy codes can become extremely large as the number of stages increases, and consequently the storage requirements for the entropy tables may very well offset the memory savings obtained by using RVQ, especially when the bit rate and/or the vector size is large. For example, consider the design of EC-RVQ codebooks where each direct-sum codebook contains 10 stages with 4 code vectors/stage. Surprisingly, more than 4 million ($4 + 4^2 + \ldots + 4^{10}$) scalar memory locations are needed to store the tables of conditional-stage entropy codes (for each direct-sum codebook). However, the number of tables can be made very small by limiting the number of previous stages upon which the conditioning is based. This can be accomplished by making a Markov-like assumption and using conditional probabilities which depend only on the last $m$ ($m < p - 1$) stages. In other words, the direct-sum codeword length $|L(j)|$ is approximated by
Obviously, since $H(J_p|J_{p-1}, \ldots, J_1) \leq H(J_p|J_{p-1}, \ldots, J_{p-m})$ for each $p = 1, 2, \ldots, P$ and $m < p-1$, it is easy to show that $H_m(J) = \sum_{p=1}^{P} H(J_p|J_{p-1}, \ldots, J_{p-m}) \geq H(J)$. When $m$ is small, the memory requirements to store these tables are relatively small, but (of course) the performance of the associated EC-RVQ is also not as good as that of the $(P-1)$th order $(m = P-1)$ EC-RVQ.

It is of particular interest to find the performance gain as a function of $m$, which will help us assess how large a value of $m$ is needed such that satisfactory performance is obtained. Such performance gain (as a function of $m$) can be estimated empirically. Figure 4 shows that the performance gain obtained when $m$ is increased, ascends rapidly to the optimal and often saturates for very small values of $m$. Using small values for $m$ has the advantage that the memory requirements can be substantially reduced.

5 Performance of EC-RVQ

In this section, experimental results are used to compare the performance and complexity of EC-RVQ with those of EC-VQ over a wide range of bit rates and vector sizes. The training set consists of six $(512 \times 512, 8$-bit) monochrome images taken from the USC database. Shifts and rotations are used to generate additional training vectors, leading to more than $200,000$ $4 \times 4$ vectors and more than $500,000$ $8 \times 8$ vectors. The image Lena, shown in Figure 5, is used for testing, and was not included in the training set. In all experiments, the objective performance measure used is the peak signal-to-quantization noise ratio (PSNR) defined by

$$PSNR = -10 \log_{10} \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (x(i,j) - \hat{x}(i,j))^2}{(N)^2(255)^2}$$
Figure 4: The rate-distortion performance of EC-RVQ (with 4 stages and 16 $4 \times 4$ vectors/stage) for the test image Lena at increasing values of $m$.

where $N \times N$ is the size of the image (assumed to be squared) and $x(i,j)$ and $\hat{x}(i,j)$ represent the original and coded values (respectively) of the pixel at the $i$th row and the $j$th column of the image.

EC-RVQ systems based on $4 \times 4$ vectors were investigated first where the EC-RVQ design algorithm with $M = 4$ and $m = 1$ was used to design a sequence of variable rate RVQ codebooks. Each codebook contained 4 stage codebooks of size 16, leading to a peak encoding rate of 1.0 bit per pixel (bpp). Likewise, the conventional EC-VQ algorithm was used to design a sequence of codebooks of size $2^{12} = 4096$. Although a moderate peak bit rate (i.e., 0.75 bpp) was used in the design of EC-VQ codebooks, the design process required well over two months of CPU time on a Sun 4 Sparc station. Figure 6 compares the distortion versus rate performance on Lena, while Figure 7 compares the encoding complexity and the memory requirements for the EC-VQ and EC-RVQ. Table 1 shows PSNR comparisons of EC-RVQ and EC-VQ.
Figure 5: The original image Lena at 8 bits/per pixel
Figure 6: The rate-distortion performance of EC-RVQ (top) and EC-VQ (bottom) for the test image Lena. The vector size is 4 x 4.

for four images (taken from the USC database) all at an output bit rate of 0.40 bits/ per pixel. EC-RVQ clearly outperforms EC-VQ in PSNR performance, encoding complexity and memory requirements. An important factor influencing the gain of EC-RVQ over EC-VQ is the very large number of direct-sum code vectors that EC-RVQ makes available, even while maintaining a low average encoding rate. The EC-VQ has a very limited codebook size (due to storage and search constraints), and the size constraint is not inactive as the theory requires.

In the second set of experiments, 4 x 4 vectors were used in the design of variable rate EC-RVQ codebooks with $M = 4$ and $m = 2$. The EC-RVQ codebooks contained 7 stage codebooks each with 16 code vectors, leading to a peak encoding rate of 1.75 bpp. Figure 8 show the PSNR performance for the EC-RVQ (at two different peak bit rates) for the test image Lena. As expected, EC-RVQ performance improves with increased peak bit rate, in spite of maintaining the same average output bit rate. It
Figure 7: The encoding complexity (top figure) and memory requirements (bottom figure) of EC-VQ (top) and EC-RVQ (bottom) for the test image Lena.
<table>
<thead>
<tr>
<th></th>
<th>EC-VQ, peak=0.75 bpp</th>
<th>EC-RVQ, peak=1.00 bpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>30.97</td>
<td>31.27</td>
</tr>
<tr>
<td>Boat</td>
<td>29.63</td>
<td>30.21</td>
</tr>
<tr>
<td>Peppers</td>
<td>31.03</td>
<td>31.32</td>
</tr>
<tr>
<td>Tiffany</td>
<td>30.08</td>
<td>30.29</td>
</tr>
</tbody>
</table>

Table 1: PSNR of EC-RVQ and EC-VQ for four images taken from the USC database. The bit rate is 0.40 bpp. The vector size is 4 x 4.

Figure 8: The rate-distortion performance of EC-RVQ for the test image Lena at two different peak bit rates (1.75 bpp for top, 1.00 bpp for bottom). The vector size is 4 x 4.
is noteworthy that EC-VQ based on high peak bit rates is not practical in general because of the large memory and complexity associated with the encoding and design procedures.

In the last set of experiments, $8 \times 8$ vector sizes were used in the design of EC-RVQ with $M = 4$ and $m = 2$ and with 7 stages codebooks of size 16. The maximum bit rate is then 0.4375 bpp. Figure 9 shows the coded image Lena at an average encoding rate of 0.1505 bpp. The PSNR is about 30 dB, and the subjective quality is rather good for a compression ratio of about 50 : 1. Practical EC-VQ systems are limited to relatively small vector sizes (typically $4 \times 4$) due to the exorbitant encoding and memory demands needed to implement such quantizers. While the EC-RVQ coding results (for $8 \times 8$ vectors) at such low bit rates cannot be compared with those of EC-VQ, they appear to be almost as good as those of more complex hybrid Subband/EC-RVQ/entropy coders reported in [21, 22].

6 Closing Remarks

The entropy-constrained RVQ introduced in this paper has many attractive features for data compression. In particular, the performance quality is among the best available to date. There also appear to be several areas where improvement can be made. One in particular is the entropy coding employed in the design algorithm. Equation (6) assumes the use of codewords that have non-integer lengths, and results in an average rate which is exactly equal to the output entropy of the EC-RVQ codebook. One can also employ (during the EC-RVQ design) an entropy coding algorithm of the entropy code that would follow the EC-RVQ. When employing a Huffman coding algorithm, both alternatives produced overall EC-VQ systems with nearly identical performance [5]. However, this may not be true in the case of EC-RVQ because the tables of conditional probabilities are usually very small (e.g. 4, 8, 16), and the average lengths of the corresponding entropy codes may not be as close to the output
Figure 9: The image Lena coded at 0.1505 bpp. The vector size is $8 \times 8$. The PSNR is 30.05 dB.
entropy. Therefore, incorporating a Huffman encoder into the EC-RVQ design algorithm may lead to a significant increase in performance when that entropy coder is used to encode the RVQ stage indices. Another important issue that relates to the entropy coding problem is the fact that, since the conditional probability distributions of the latter stages are usually very skewed, other entropy coding techniques (such as arithmetic coding) may perform better than Huffman-based techniques. Experiments where both a Huffman encoder and an arithmetic encoder are separately incorporated into the EC-RVQ design algorithm are presently being investigated.

Another possible area for improvement is in the entropy coding structure. The present design algorithm is based on static entropy codes. However, with both the size and the number of the tables being relatively small, the possibility of adaptive entropy coding exists. This is another variation of the system presently being investigated.

Finally, we point out that EC-RVQ has some potential advantages in terms of channel insensitivity characteristics. Fixed rate RVQ tends to be less sensitive to channel errors than conventional VQ. A bit error could be disastrous for a conventional VQ, but is usually less serious when an RVQ is used. This nice property of RVQ seems to be lost when an EC-RVQ is used because a bit error in one of the stage codewords will very likely propagate through the subsequent stages, and will prevent the RVQ decoder from correctly decoding the variable length codewords of the remaining stages. However, the EC-RVQ variable length codewords become less sensitive to channel errors if we were to protect those variable length codewords of the first few stages.
List of Figures

1 A P-stage residual vector quantizer ........................................... 4
2 The EC-RVQ Structure .............................................................. 12
3 The EC-RVQ design algorithm ................................................... 15
4 The rate-distortion performance of EC-RVQ (with 4 stages and 16 \(4 \times 4\) vectors/stage) for the test image Lena at increasing values of \(m\) .................................................. 18
5 The original image Lena at 8 bits/per pixel .................................. 19
6 The rate-distortion performance of EC-RVQ (top) and EC-VQ (bottom) for the test image Lena. The vector size is \(4 \times 4\) ........................................... 20
7 The encoding complexity (top figure) and memory requirements (bottom figure) of EC-VQ (top) and EC-RVQ (bottom) for the test image Lena ......................................................... 21
8 The rate-distortion performance of EC-RVQ for the test image Lena at two different peak bit rates (1.75 bpp for top, 1.00 bpp for bottom). The vector size is \(4 \times 4\) ......................................................... 22
9 The image Lena coded at 0.1505 bpp. The vector size is \(8 \times 8\). The PSNR is 30.05 dB ......................................................... 24
List of Tables

1. PSNR of EC-RVQ and EC-VQ for four images taken from the USC database. The bit rate is 0.40 bpp. The vector size is $4 \times 4$. 22
References


NSF Annual Report, Year 2
Project Title: Dynamic Multirate Systems for Image and Video Compression
Grant MIP-9116113

Mark J. T. Smith
Georgia Institute of Technology
School of Electrical Engineering
Atlanta, Georgia 30332

This document reports on progress related to the second year of research under NSF Grant MIP-9116113. The research focuses on new directions in the areas of multirate decomposition (subband coding) and vector quantization with particular application to image and video compression. During the first year of funding, several innovative approaches to the problem were developed, as reported in the first annual report: FIR spatially adaptive filter banks; IIR spatially adaptive filter banks; high dimension vector quantization; and entropy-constrained residual vector quantization. During this second year of funding, we have continued our study of these approaches and in some cases improved upon them. The evolution of this work over the past 12 months has resulted in some new variations and approaches, which are summarized briefly in the body of this report. A detailed description of the work is provided through the journal and conference papers that are included in the appendix.

Spatially Adaptive Filter Banks: In Spring 1992, we introduced the concept and theory of FIR filter banks with the property of on-line adaptivity and exact reconstruction [1]. Last year, we introduced several classes of adaptive filter banks (some FIR and some IIR) and showed that they had the potential to improve performance in a subband coder [2], [3], [4], [5]. During this year we continued our examination of these techniques. It is clear to us that significant gains are possible, but several issues must be resolved before the full magnitude of these gains can be realized. The first issue, is the determination of desirable characteristics for the filter bank sets that are used in the adaptive system. This last year we used a strategy based on switching between filters with good step response and magnitude response characteristics. This led to some improvement as is shown in reference [5]. This year we are examining a superior strategy based on switching between minimum-phase and maximum-phase filter bank sets. The capability to do this new form of adaptivity is a result of a new post-filtering theory. This work is described in reference [6], which is included in the appendix as part of this report.

A second issue is the design of an effective adaptation control network. This network tracks spatially the characteristics of the image as it is being coded and matches the best of the predesigned filter bank sets with the current input region being processed. Presently we do this by sending side information. However this can be done via backward adaptation, thereby avoiding side information altogether. This is still under investigation.

Other issues include: the behavior of the filter bank during the transition between filter sets; ease of implementation; and constraints of how often one can update the
coefficients. At this point, the new post-filter approach [6] seems to be attractive by all measures.

**Entropy-Constrained RVQ:** Residual vector quantization has the nice property of reducing both computation and memory requirements, but has the undesirable effect of degradation in performance relative to that of conventional vector quantization (VQ). In our last annual report, we introduced entropy-constrained RVQ and showed that this drawback can be overcome by exploiting the source entropy explicitly. During this year, we developed the mathematics supporting optimality of the EC-RVQ design method. This theory is given in a journal paper [7], which was submitted to the IEEE Transactions on Information Theory.

**Alphabet and Entropy Constrained RVQ:** A new variation of the entropy-constrained RVQ work, which we studied this year, involves placing a constraint on the output alphabet of the RVQ. We call this variation *alphabet and entropy constrained RVQ* (AEC-RVQ). Optimality conditions for the design of the system have been derived and a design algorithm has been developed and evaluated. Although the addition of an alphabet constraint results in some loss in performance, the loss is very small. What is important is that there is a tremendous gain in memory savings for storage of the codebook. Compared to the EC-RVQ system discussed in [7], the AEC-RVQ only requires 1/25 of the memory. This work is discussed in reference [8] and is included as part of this report.

**Adaptive EC-RVQ and AEC-RVQ:** It is well known in the community that if a VQ is designed with the test image in the training set, the performance results are significantly improved. Adaptive VQ methods have been considered previously (by Goldberg) that consist of designing a VQ with the test image included in the training set and then transmitting the codebook as side information. The major problem with this approach is that the side information involved in sending the VQ codebook is significant. Nonetheless, Goldberg has shown that some gains can be achieved in spite of this.

During this year we examined the idea of adaptive EC-RVQ and AEC-RVQ. Such an approach would seem very attractive for these systems because they are so codebook-memory efficient compared to conventional VQ. However, the advantages are even greater than this because the codebooks don't have to be sent at all. Only the vector probabilities need to be known at both sides. Thus forward adaptive and backward adaptive EC-RVQ and AEC-RVQ schemes were developed and tested. It was shown that about a twenty percent improvement could be achieved in performance by using an adaptive approach. This work was presented at the AIAA Conference in November [9], a copy of the paper is included in the appendix as part of this report.

**Subband Coding:** There are many subband image coding schemes reported in the literature (many of them going under the alias of wavelet transform coding). This last year, we looked at combining efficient subband analysis/synthesis systems with EC-RVQ. This work is discussed in reference [10]. This investigation was a straightforward experimental study. The results, however, were among the best reported at the time.

In the next phase of this work, we considered an innovative approach to the subband coding problem. Specifically, we considered the construction of a new subband coder,
where the quantizers and entropy coder were designed jointly to be optimal for a given analysis/synthesis system and input data set. Thus far, these quantizers are only scalar quantizers. This approach is fundamentally different from designing optimal analysis/synthesis systems, quantizers, and entropy coders in isolation, as has been the practice previously. We call this new system entropy-constrained subband coding or EC-SBC. The theory underlying this method is discussed in reference [11], and is included as part of this report. A paper addressing the design of systems based on this theory is in preparation and will be included in the next annual report. The EC-SBC is given the best results we have had to date. This is very encouraging because there are many additional feature that we have not yet included that we know will further improve performance, such as using vectors instead of scalar quantizers, using adaptive filter banks, and using adaptive space-frequency resolution filter banks.

Remarks Regarding Impact of Work

The work in spatially adaptive filter banks is progressing nicely and has great potential benefit. This is because its contributions are additive. In other words, adaptive filter banks can be used in place of fixed filter banks in the best existing subband coding and the performance results are guaranteed to improve or remain the same. Moreover, adaptive filter banks offer the only method known to the PI to remove ringing distortion in subband coders at low bit rates.

VQ schemes used in conjunction with subband and transform coding yield some of the best coding methods presently known. The work in entropy-constrained RVQ introduced here represents the first VQ-system, to our knowledge, that achieves this high level of competitiveness with subband and transform coding when applied DIRECTLY to the image input. The AEC-RVQ and adaptive variants of this system allow for greater memory efficiency and robustness. These new methods are powerful in their own right, but can also be used in conjunction with other coding methods.

Among the methods we are presently exploring, the new EC-SBC system represents the approach with the greatest potential. In all the comparisons and evaluations we have performed with competing systems, EC-SBC gives the best performance both in terms of PSNR and subjective quality. We are presently examining this coder for application to telemedicine. Telemedicine is a very exciting application area in which to work. The benefits of telemedicine are expected to revolutionize health-care in rural America.

We are also making progress in extending these ideas to the problem of video coding. We will have more to say about video coding applications in the next annual report.

References


3. J. Arrowood and M. Smith, "Exact Reconstruction Analysis/Synthesis Filter Banks with Time-Varying Filters," Proceedings of IEEE Int. Conf. on Acoustics, Speech,


Time-Varying Analysis-Synthesis Systems Based on Filter banks and Post Filtering

Iraj Sodagar
Kambiz Nayebi
Thomas P. Barnwell, III
Mark J. T. Smith

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250

Abstract

Perfect reconstruction (PR) time-varying analysis-synthesis filter banks are those in which the filters are allowed to change from one set of PR filter banks to another as the input signal is being processed. Such systems have the property that, in the absence of coding, they faithfully reconstruct every sample of the input. Various methods have been reported for the time-varying filter bank design, all assume the conventional structure for time-varying filter banks. This constraint results in different limitations in each method. This paper introduces a new structure for exact reconstructing time-varying analysis-synthesis filter banks. This structure consists of the conventional filter bank followed by a time-varying post filter. The new method requires neither the redesign of the analysis sections, nor the use of any intermediate analysis filters during transition periods. It provides a simple and elegant procedure for designing time-varying filter banks without the disadvantages of the previous methods.

The address of the contact author:
Iraj Sodagar
School of Electrical and Computer Engineering
Georgia Tech,
Atlanta, GA 30332-0250
Tel:(404) 894-2977 ext-4
Fax:(404)894-8363
e-mail: iraj@eedsp.gatech.edu

EDICS Category: 2.4.2 and 2.4.4
1 Introduction

Linear time-Frequency decompositions such as those based on uniform band and tree structured filter banks have found many applications in such areas as speech, image and video coding. In particular, maximally decimated finite impulse response (FIR) analysis-synthesis systems capable of exactly reconstructing the input signal have received high visibility in the literature. References such as [1, 2, 3, 4] cover the main issues related to the classical filter bank theory and design. The underlying basis functions of these conventional transforms are fixed in time and define a specific tiling of the time-frequency plane. The time-frequency energy distribution of the signals is typically encountered in practice, however, often changes with time. Thus in this sense, the conventional linear time-frequency transform paradigm is fundamentally mismatched to the signals of interest. A more flexible and accurate approach is obtained if the basis functions of the transform are allowed to adapt to the signal properties. In this scenario, the time-frequency tiling of the transform can be changed from good frequency localization to good time localization and vice versa. In this general context, Coifman and Wickerhauser [5] introduced the idea of finding the "best" wavelet packet basis for a given signal while Ramchandran and Vetterli [6] reported an algorithm to code the nonoverlapping portions of a signal with the best wavelet packet basis in the rate-distortion sense. However, the concept of time-varying overlapping block transforms using analysis-synthesis filter banks is quite new.

Analysis-synthesis filter banks can be time-varying in different ways. For example, it is possible to vary the analysis and/or synthesis filters, the down-up sampling rates, or the number of bands in the system. This is illustrated by the block diagram shown in Figure (1). Using such time-varying systems, the basis vectors and the tiling of the time-frequency plane can be changed at any time. A simple form of time-varying filter bank is achieved by changing the filters of an analysis-synthesis system among a finite number of choices. Even if all the analysis and synthesis filters are perfect reconstruction (PR) sets, exact reconstruction will normally not be achieved when one switches among those sets. This is because during the transition, some input samples used by the second synthesis filters were generated by the first analysis filters which results in reconstruction errors.

Various methods have been reported for the time-varying filter bank design. Nayebi et al. [7] introduced one class of time-varying filter banks. The ideas underlying their method were later devel-
Figure 1: The time-varying filter bank structure with time-varying filters and time-dependent down/up samplers.

opened and extended to a more general case in which it was also shown that the number of frequency bands could also be made adaptive [8, 9, 10]. In this approach, the analysis filter bank may be switched to another set of analysis filters arbitrarily. This means that the basis vectors and the tiling of the time-frequency plane can be changed instantaneously. If exact reconstruction is to be achieved in this context, the two analysis filter banks must be interrelated. To achieve perfect reconstruction at each time in the transition period, a new synthesis section is designed to insure proper reconstruction. Since such synthesis filter banks will not always exist for any arbitrary set of analysis filters, the analysis filter banks must be redesigned simultaneously to make the time-varying system perfect reconstruction. One alternative solution is to use a few sets of intermediate analysis-synthesis filter banks during transition [11, 12, 13]. These intermediate analysis transforms are chosen such that the PR property is ensured during the transition period. The characteristics of these analysis transforms are not easy to control and typically the intermediate filters are not well-behaved. In [14], time-varying filter banks are obtained using time-reversed filters. This method does not always result in stable systems. Size-limited filter banks [15, 16] are a special class of time-varying filter banks and can be studied in this context.

All previous methods [8, 9, 10, 11, 12, 13, 14] assume the conventional structure for time-varying filter banks similar to the one which is used for the time-invariant case. More specifically, the synthesis filters have the same length in the transition periods as in the time-invariant periods. This constraint results in different limitations in each method [17]. In this paper, a new structure of the time-varying analysis-synthesis structures with some important features is introduced [18, 19]. First, the
Figure 2: Block diagram of a switching 2-band/3-band filter bank.

analysis section may be switched from one filter bank to another instantaneously while preserving the PR property at all time including transition periods. Therefore, this method does not require any intermediate analysis transform. Second, this method does not require any redesigning of the analysis filters as in the earlier approach. The new structure allows for a simple implementation and also provides a simple procedure for designing the time-varying filter banks. The nature of the formulation allows for the simple construction of a large set of systems, each with different characteristics.

The new formulation is based on representing the time-varying filter bank as a conventional filter bank with time-varying coefficients followed by a time-varying post filter. The post filter provides exact reconstruction during transition periods, while it operates as a constant delay otherwise.

In the following section, the notion of the transition period is discussed and examined. Next, we present the post filter derivation, the necessary conditions for its existence and examine its performance. Prior to the concluding remarks, three design examples of time-varying filter banks are given to illustrate the use and effectiveness of the post filter approach.

**Notation:** For clarity and convenience of the presentation, a consistent notation is used throughout the paper as described next. Boldface letters are used to denote matrices and vectors. Several vectors and matrices will appear repeatedly in the text. The vector $\mathbf{x}_N(n)$ is defined to be $[x(n), x(n-1), \ldots, x(n-N-1)]^T$ of length $N$ where $n$ is the time index. $\mathbf{A}_{M \times N}$ is a matrix of size $M \times N$
and \( B_M \) is a square matrix of size \( M \). Superscript \( T \) denotes matrix (or vector) transposition. The determinant and inverse of matrix \( A \) are given by \( \det(A) \) and \( A^{-1} \) respectively. \([A]_{ij}^T\) is the matrix consisting of rows \( i, i+1, \ldots, j \) of matrix \( A \) where row and column numbering starts from zero. In this notation, if \( i \) is a negative integer and/or \( j \) is larger than the size of the matrix, the appropriate number of zero row vectors is added. For example, if \( A = [1, 2, 3, 4]^T \), then \([A]_{1}^{-1} = [0, 1, 2]^T\), \([A]_{3} = [2, 3, 4]^T\) and \([A]_{4} = [3, 4, 0]^T\). \( \|\mathbf{x}\| \) is the \( L_2 \) norm of vector \( \mathbf{x} \) while \( \|A\| \) is a row vector consisting of the \( L_2 \) norm of the corresponding columns of matrix \( A \). \( \mathbf{0} \) and \( \mathbf{I} \) are zero and identity matrices respectively. Square brackets denote closed integer intervals while parenthesis define open integer intervals. Thus for example, the interval \([n_1, n_2)\) consists of samples \( n_1, n_1 + 1, \ldots, n_2 - 1 \).

2 Time-Varying Filter Banks

A time-varying analysis/synthesis filter bank is a system in which the analysis filters, the synthesis filters, the number of bands, the decimation rates, and the frequency coverage of the bands are changed (in part or in total) in time (Figure 1). A simple form of this filter bank is achieved by changing the analysis-synthesis filters from among a number of pre-calculated PR filter bank sets. Each analysis-synthesis filter set can be considered as a time-invariant filter bank and assumed to provide perfect reconstruction. Figure (2) shows the block diagram of a time-varying system which includes one 2-band and one 3-band filter bank.

This paper studies the time-varying filter banks which are obtained by switching among a set of time-invariant filter banks. Such a time-varying filter bank operates in two different time intervals: time-invariant periods and transition periods. A time-invariant period is defined to be the period in which the synthesis filter inputs are generated from only one analysis set. Therefore, the corresponding synthesis set provides the perfect reconstruction within this period. The transition period is defined to be the period in which the synthesis filter inputs are generated from more than one set of analysis filters. It occurs during the switching from one filter bank set to another. In such systems, the analysis-synthesis system delay must always stay constant to avoid missing or repeating any sample at the system output.
2.1 Time-Domain Formulation

In order to analyze and design time-varying filter banks, we use the time-varying extension of the time-domain formulation reported by Nayebi et al [20]. The time-varying version of this formulation is presented in Appendix. Other proposed time-domain formulations [21, 22] can also be extended to the time-varying case, following the discussion presented in this paper. Consider that the impulse response of a time-varying analysis-synthesis system at time \( n \), \( z(n) = [z(n, n), z(n, n-1), \ldots, z(n, n-2N(n)+1)] \) can be described as:

\[
z(n) = A(n)s(n) \tag{1}
\]

where \( A(n) \) is the matrix that contains the analysis filter coefficients which produce the existing samples in the synthesis section at time \( n \) and \( s(n) \) consists of the synthesis filter coefficients at that moment. \( N(n) \) denotes the maximum length of the filters at time \( n \). A PR time-varying filter bank is obtained if \( z(n) = b(n) \) for all values of \( n \), where vector \( b(n) \) is the ideal impulse response at time \( n \) (see Appendix A). The average signal-to-reconstruction-noise ratio (ASNR) [10] of a time-varying filter bank in the interval \( [n_0, n_0 + L] \) is defined as

\[
\text{ASNR}(n_0, L) = -10 \log_{10} \left[ \frac{1}{L} \sum_{n=n_0}^{n_0+L-1} \| z(n) - b(n) \|^2 \right]. \tag{2}
\]

The ASNR can be used as a measure to compare the reconstruction quality of different designs for a desired time-varying filter bank. For a time-varying filter bank consisting of \( P \) different time-invariant filter banks, it is convenient to define the ASNR matrix as

\[
\text{ASNR}_{ij} = \text{ASNR}(n_{ij}, L_{ij}) \tag{3}
\]

where \( [n_{ii}, L_{ii}] \) is one time-invariant period of the \( i^{th} \) filter bank and \( [n_{ij}, L_{ij}] \) is the transition period during switching from the \( i^{th} \) filter bank to the \( j^{th} \) filter bank, for \( i \in [1, P] \) and \( j \in [1, P] \) and \( i \neq j \). So each entry in \( \text{ASNR} \) shows the quality of reconstruction in one distinct period of the time-varying filter bank. As an example, a time-varying filter bank based on \( P \) independently designed perfect reconstruction time-invariant filter banks has a \( P \times P \) ASNR matrix with infinity valued entries on the main diagonal and finite valued entries elsewhere.
Figure 3: Frequency magnitude characteristics of two independent 2-band and 3-band near PR filter banks. The 2-band filter bank has 8-tap filters with the responses shown by the solid lines. The 3-band filter bank has 12-tap filters with the responses shown by the dashed lines.

Figure 4: The time-varying impulse response for direct switching between the 2-band and the 3-band system. The filter bank is switched from 2-band to 3-band at time $n = 0$ and switched back at time $n = 13$. (a) surface plot (b) contour plot.
2.2 The Transition Period

Changing from one arbitrary filter bank set to another independently designed filter bank set without using any intermediate filters is called direct switching. In this mode of operation, the first analysis section is replaced by the second one at the switching time \( n_0 \). The first synthesis section is replaced with the second one at the same time \( n_0 \) or with a delay. Direct switching, although is potentially attractive from an applications perspective, usually results in a substantial amount of reconstruction distortion during the transition period. This behavior results from the fact that during transition none of the synthesis filters satisfy the exact reconstruction conditions. Figure (2) shows an example of a direct switching filter bank. Consider the specific case where the 2-band system is a PR filter bank with 8-tap analysis/synthesis filters and a delay of 7 samples. Two additional delays are added to the filters of the 2-band system to maintain an overall delay of 11 samples. The 3-band system is a PR filter bank with 12-tap analysis/synthesis filters and a delay of 11 samples. The spectral magnitude response characteristics of both system filters are shown in Figure (3). Figure (4) shows the time-varying impulse response of the above system around the transition periods. In this figure, \( z(n, m) \) is the response of the system at time \( n \) to the unit input at time \( m \). For a PR system, \( z(n, m) \) has a height of one along the diagonal and zero everywhere else in the \((m, n)\)-plane. As is shown, the time-varying filter bank is perfect reconstruction before and after but not during the transition periods. In this case, each switching operation generates a distortion with 8 samples duration. One way to reduce the distortion is to switch the synthesis filters with an appropriate delay with respect to the analysis switching time [10]. This is because at the beginning of the transition period, all input samples of the synthesis section but one are generated by the first analysis filters and they are gradually replaced by the new samples generated by the second analysis filter bank during the transition period. Therefore the first synthesis filters generate less distortion in the beginning samples of the transition period while the second synthesis filters have a better performance for the ending samples of the transition [10].

Now consider switching from an \( M_1 \)-band system with \( N_1 \)-tap filters to an \( M_2 \)-band system with \( N_2 \)-tap filters, assuming the delay in both system is equal. The length of the synthesis filters \( (N_T) \), the number of bands \( (M_T) \), and the minimum decimation \( (R_T) \) of the transient period are given in Table (1). To keep the system maximally decimated, the number of discarded samples between the old and
Synthesis Filters' Length
# Bands
Minimum Decimation/Interpolation ratio
Length of Transition Period
# Transition States/# Required LS Synthesis Sections

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_T = \max(N_1, N_2)$</td>
<td></td>
</tr>
<tr>
<td>$M_T = \max(M_1, M_2)$</td>
<td></td>
</tr>
<tr>
<td>$R_T = \min(M_1, M_2)$</td>
<td></td>
</tr>
<tr>
<td>$N_2 - R_T$</td>
<td>$\frac{N_2 - R_T}{R_T}$</td>
</tr>
</tbody>
</table>

Table 1: The time-varying filter bank parameters during transition period. $N_1$ and $N_2$ are the length of filters and, $M_1$ and $M_2$ are the decimation ratio in the first and second maximally decimated filter banks respectively.

new nonzero samples is chosen to be $M_1 - 1$. The transient period has a length of $N_2 - R_T$ samples. Note that the length of the transient period is different for switching back from the $M_2$-band system to the $M_1$-band system if $N_2 \neq N_1$. These transient periods can have the same length if enough delay is added to the shorter length filters [10].

One way to design a time-varying PR filter bank is to redesign the analysis and synthesis sections for the time-varying case [10]. In time-varying filter banks, the number of possible equations depends on the number of analysis banks and their permissible application orders. By determining all permissible sequences of the analysis banks, all possible states of the system can be determined. For each state, the perfect reconstruction conditions are written as a matrix equation. For given analysis sets, these equations are usually overconstrained. Using the independently designed analysis sections for the regular states, one can approximately solve the equations for each transient state using the least squares (LS) method. Although the reconstruction quality of the system improves by using the LS solutions, perfect reconstruction is generally not achieved. For a perfect reconstruction design, the independent filter banks are used as the starting point in a design procedure involving optimization [10]. The drawback of this approach is that no explicit control is available to utilize independently designed sets of PR filter banks.

3 Post Filtering

As is shown in Section 2, the output of the time-varying filter bank is distorted during transition periods. This output is the result of filtering input signal with a time-varying filter. To reconstruct the original signal, we propose a time-varying post filtering to undo the filtering operation of the time-varying filter bank during the transition period. In this section, first the post filter impulse response
is derived. Then, the necessary and sufficient conditions for existence of the post filter are studied.

### 3.1 Post Filter Derivation

Assume at time $n_0$ the time-varying filter bank is switched from the first filter bank to the second one. If the length of the transition period is $L$ samples, the output of the filter bank in the interval $[n_0, n_0 + L - 1]$ is distorted because of switching. The job of the post filter is to remove this distortion.

The block diagram of such a system is shown in Figure (5). In this figure, $z(n)$ and $y(n)$ are the filter bank and post filter time-varying impulse responses respectively. Figure (6) shows the timing diagram for the signals around the transition period. If the delays of the filter bank and the post filters are denoted $\Delta$ and $\Theta$ respectively, the filter bank output $\hat{x}(n)$ can be written as

$$
\hat{x}(n) = \begin{cases} 
\text{Distorted} & \text{if } n_0 \leq n < n_0 + L \\
\hat{x}(n - \Delta) & \text{otherwise}
\end{cases},
$$

resulting in the post filter output

$$
\hat{x}(n) = x(n - \Theta - \Delta).
$$

The input/output relation of the time-varying filter bank during the transition period can be written as

$$
\hat{x}(n) = z^T(n) \, x_f(n)
$$

where $x_f(n)$ is the input vector at time $n$:

$$
x_f(n) = [x(n), x(n-1), x(n-2), \ldots, x(n-I+1)]^T.
$$

The column vector $z(n)$ represents the time-varying impulse response of the filter bank at time $n$ as defined in equation (1). and $I$ denotes the length of the impulse response $z(n)$. If the transition impulse response matrix is defined to be
Figure 6: Timing diagram illustration of the transition period and the delays associated with the filter bank output $\hat{x}(n)$ and the post filter output $\hat{\hat{x}}(n)$.

\[
Z = \begin{bmatrix}
  z(n_0 + L - 1) & O & O \\
  O & z(n_0 + L - 2) & O \\
  O & O & \vdots \\
  \vdots & \vdots & \ddots & O \\
  O & O & \cdots & z(n_0)
\end{bmatrix},
\]

(7)

then the input/output relation of the filter bank in the transition period can be described as

\[
\hat{x}_L(n_0 + L - 1) = Z^T x_K(n_0 + L - 1)
\]

(8)

where $Z$ is a $K \times L$ matrix and $K = I + L - 1$. In equation (8), the $I - \Delta - 1$ samples before and $\Delta$ samples after the transition period are used to evaluate the output. The above intervals are called the tail and head of the transition period respectively and are shown in Figure (6). As the first and second filter banks are PR, the tail and head samples are exactly reconstructed. Therefore, equation (8) can be expanded to the following block transform

\[
\hat{x}_K(n_0 + L + \Delta - 1) = W^T x_K(n_0 + L - 1)
\]

(9)

where
In order to remove distortion, the inverse block transform must be applied to the filter bank output

$$\hat{x}_K(n_0 + L + \Delta + \Theta - 1) = V^T \hat{x}_K(n_0 + L + \Delta - 1)$$

where $V = W^{-1}$ is the inverse transform. If $W$ is written as

$$W = \begin{bmatrix} I_\Delta & Z_a & 0 \\ 0 & Z_t & 0 \\ 0 & Z_b & I_{I-\Delta-1} \end{bmatrix},$$

it is easy to see that

$$V = W^{-1} = \begin{bmatrix} I_\Delta & -Z_a Z_t^{-1} & 0 \\ 0 & Z_t^{-1} & 0 \\ 0 & -Z_b Z_t^{-1} & I_{I-\Delta-1} \end{bmatrix}.$$ 

Therefore the input/output relationship of the post filter during the transition period is defined by

$$\hat{x}_L(n_0 + L + \Theta - 1) = Y^T \hat{x}_K(n_0 + L + \Delta - 1).$$

In this equation, $Y$ is the post filter time-varying impulse response in the transition period which is defined as

$$Y = \begin{bmatrix} -Z_a Z_t^{-1} \\ Z_t^{-1} \\ -Z_b Z_t^{-1} \end{bmatrix}.$$ 

From (14), it is obvious that the condition for casual post filtering is

$$\Theta \geq L + \Delta - 1.$$ 

Note that $VV = WV = IK$, so these block transforms can commute i.e the filter bank-post filter combination is commutative.

We have shown that to remove the distortion in a transition period, it is necessary that the tail and head of the transition periods be perfectly reconstructed prior to post filtering. Now, consider a second case where two consecutive switchings occur in the analysis filter bank. Each switching causes a transition period at the filter bank output. If the second switching occurs with sufficient delay relative to the first switching, the heads and tails of each transition period are perfectly reconstructed. In this case, each transition period can be recovered by an independently designed post filter. Even if the tail of the second transition period contains some distorted samples of the first transition, each of the transition periods can be reconstructed using one separately designed post filter. But here, in spite of the first case, the post filters can be considered as the first and second iterations of a time-varying
block IIR filter. The stability of this IIR filter must be studied if tail overlapping occurs frequently and without any interruption. Finally, consider the third the case in which the head of the first transition period contains some distorted samples of the second transition. Here, separate post filtering can not reconstruct the samples of the transition periods. To retrieve these samples, however, one can define a new transition period consisting of both transition periods and apply one post filter to the new transition period.

Example: To illustrate the post filter derivation, consider switching from one 2-band system to another 2-band system, both with 4-tap filters and delay of 3 samples. The transition period has a length of 2 samples and the length of the impulse response in the transition period is 7:

\[
z(0) = [z(0, 0), z(0, -1), \ldots, z(0, -5), 0]^T,
\]

\[
z(1) = [0, z(1, -1), z(1, -2), \ldots, z(1, -6)]^T.
\]

Therefore the transition impulse response matrix has a size of \(8 \times 2\)

\[
Z = \begin{bmatrix}
0 & 0 \\
z(1, -1) & z(0, 0) \\
z(1, -2) & z(0, -1) \\
z(1, -3) & z(0, -2) \\
z(1, -4) & z(0, -3) \\
z(1, -5) & z(0, -4) \\
z(1, -6) & z(0, -5) \\
0 & 0
\end{bmatrix}
\]

which is divided to three submatrices

\[
Z_a = \begin{bmatrix}
0 & 0 \\
z(1, -1) & z(0, 0) \\
z(1, -2) & z(0, -1)
\end{bmatrix}
\]

\[
Z_t = \begin{bmatrix}
z(1, -3) & z(0, -2) \\
z(1, -4) & z(0, -3)
\end{bmatrix}
\]

\[
Z_b = \begin{bmatrix}
z(1, -5) & z(0, -4) \\
z(1, -6) & z(0, -5)
\end{bmatrix}
\]

To calculate the post filter, \(Z_t^{-1}\) is obtained

\[
D = \det(Z_t) = z(0, -3)z(1, -3) - z(0, -2)z(1, -4) \neq 0
\]

\[
Z_t^{-1} = \frac{1}{D} \begin{bmatrix}
z(0, -3) & -z(0, -2) \\
-z(1, -4) & z(1, -3)
\end{bmatrix}
\]

\(Y\) is calculated using (15):

\[
Y = \frac{1}{D} \begin{bmatrix}
- \begin{bmatrix}
z(1, -3) & 0 \\
z(1, -4) & z(0, -3)
\end{bmatrix} & \begin{bmatrix}
z(0, -3) & -z(0, -2) \\
-z(1, -4) & z(1, -3)
\end{bmatrix}
\end{bmatrix}
\]

\[
- \begin{bmatrix}
z(1, -5) & z(0, -4) \\
z(1, -6) & z(0, -5)
\end{bmatrix} & \begin{bmatrix}
z(0, -3) & -z(0, -2) \\
-z(1, -4) & z(1, -3)
\end{bmatrix}
\end{bmatrix}
\]
3.2 Existence of the Post Filter

In this section, the necessary conditions for post filtering reconstruction are investigated. Specifically, the necessary conditions for existence of the post filter is derived in terms of analysis and synthesis filters of time-invariant filter banks.

As is shown in Section 3.1, the post filter exists if $Z_t$ has an inverse. From (1) of Appendix, each column of the transition impulse response matrix $Z$ can be expressed as matrix products:

$$Z = \begin{bmatrix}
\begin{bmatrix} A(n_0 + L - 1)s(n_0 + L - 1) \\
O \\
O \\
\vdots \\
O
\end{bmatrix} & O & O \\
\begin{bmatrix} A(n_0 + L - 2)s(n_0 + L - 2) \\
O \\
O \\
\vdots \\
O
\end{bmatrix} & \ddots & \ddots \\
\begin{bmatrix} A(n_0)s(n_0) \\
O
\end{bmatrix}
\end{bmatrix}.$$  \hfill (17)

From equation (56), it is obvious that the last $(N_T - 1)M_T$ columns of $A(n)$ are the same as the first $(N_T - 1)M_T$ columns of $A(n - 1)$ with one additional zero row vector at the top. Therefore, equation (17) can be written as

$$Z = \Psi S$$ \hfill (18)

where

$$\Psi = \begin{bmatrix}
\begin{bmatrix} P(n_0 + L - 1)^T A(n_0 + L - 1) \\
O_{1 \times M_T} \\
O_{1 \times M_T} \\
\vdots \\
O_{1 \times M_T}
\end{bmatrix} & O_{1 \times M_T} & \ldots & O_{1 \times M_T} \\
\begin{bmatrix} P(n_0 + L - 2)^T A(n_0 + L - 2) \\
O_{1 \times M_T} \\
O_{1 \times M_T} \\
\vdots \\
O_{1 \times M_T}
\end{bmatrix} & \ddots & \ddots & \vdots \\
\begin{bmatrix} P(n - N(n) + 1)^T A(n - N(n) + 1) \\
O_{1 \times M_T} \\
O_{1 \times M_T} \\
\vdots \\
O_{1 \times M_T}
\end{bmatrix}
\end{bmatrix},$$ \hfill (19)

$$S = \begin{bmatrix}
\begin{bmatrix} s(n_0 + L - 1) \\
O_{M_T \times 1} \\
O_{M_T \times 1} \\
\vdots \\
O_{M_T \times 1}
\end{bmatrix} & O_{M_T \times 1} & O_{M_T \times 1} \\
\begin{bmatrix} s(n_0 + L - 2) \\
O_{M_T \times 1} \\
O_{M_T \times 1} \\
\vdots \\
O_{M_T \times 1}
\end{bmatrix} & \ddots & \ddots \\
\begin{bmatrix} s(n_0) \\
O_{M_T \times 1} \\
O_{M_T \times 1} \\
\vdots \\
O_{M_T \times 1}
\end{bmatrix}
\end{bmatrix}.$$ \hfill (20)

$\Psi$ is a $(2N_T + L - 2) \times (N_T + L - 1)M_T$ matrix containing only the analysis filters while $S$ is a matrix of size $(N_T + L - 1)M_T \times L$, consisting of the synthesis filter coefficients. Therefore based on the analysis and synthesis filters, $Z_t = [Z]_{\Delta + L - 1}^\Delta$ can be expressed as

$$Z_t = [\Psi]_{\Delta + L - 1}^\Delta S = \Psi L S.$$ \hfill (21)
In order for $Z_t$ to be invertible, it is necessary (but not sufficient) that $\Psi_L$ and $S$ be full rank matrices. $\Psi_L$ consists of some coefficients of the first and the second analysis sections. As these analysis sections are defined by the required properties of the first and second filter banks, $\Psi_L$ is fixed. Therefore a filter bank is *switchable* to another filter bank if the corresponding $\Psi_L$ is a full rank matrix. Each column in $S$ represents the synthesis section at one moment of the transition period. In spite of the analysis section, the synthesis section of the time-varying filter bank can change during the transition period. Therefore, by properly designing the synthesis section, both $S$ and $Z_t$ will be full rank. In the next section, we present two methods to do this.

The above conditions reflect the fact that the analysis and synthesis transforms of the time-varying filter bank must be full rank transforms during the transition period. To better visualize this, consider the doubly infinite transform matrix [23] for a time-varying analysis filter bank as

$$\Pi = \begin{bmatrix}
\vdots \\
\Lambda(n+3)P(n+3) \\
\Lambda(n+2)P(n+2) \\
\Lambda(n+1)P(n+1) \\
\Lambda(n)P(n) \\
\Lambda(n-1)P(n-1) \\
\Lambda(n-2)P(n-2) \\
\Lambda(n-3)P(n-3) \\
\vdots 
\end{bmatrix}. \tag{22}$$

$\Pi$ can be written by extending $\Psi^T$ to an infinite size matrix. In the similar way, by extending $S^T$, a doubly infinite matrix can be obtained as the inverse transform (call it $\Sigma$). The input of the synthesis filters at time $n$ is obtained by multiplying the $M(n)$ rows of $\Pi$ (denoted by time index $n$) by the input vector $x_N(n)$. The output of synthesis section is obtained from the inner product of the row of $\Sigma$ denoted by index $n$ and the vector consisting of the inputs of synthesis section of times $n, n-1, \ldots, n-N+1$. Therefore the input/output relationship of the filter bank can be written as

$$\hat{x}_\infty = \Sigma \Pi x_\infty \tag{23}$$

where $x_\infty$ and $\hat{x}_\infty$ are doubly infinite input and output vectors. In fact, equation (53) of Appendix shows the above relation just for time $n$. For a PR filter bank, the output at time $n$ must be equal to $x(n-D)$. To provide perfect reconstruction for any input block of size $L$, it is necessary and sufficient that
1. The columns of $\Pi$ which are multiplied by the corresponding input block constitute a full rank matrix.

2. The columns of $\Sigma$ which are multiplied by the above transform outputs constitute a full rank matrix.

3. The column space of the first transform be identical to the row space of the second transform.

If the analysis sections are switched at time $n = n_0$ and the transition period has a length of $L$, $\mathbf{x}_L(n_0 + L - \Delta) = [x(n_0 + L - \Delta), x(n_0 + L - \Delta - 1), \ldots, x(n_0 - \Delta + 1)]^T$ consists of the input samples that are not exactly reconstructed. The columns of $\Pi$ and $\Sigma$, which are multiplied by this vector and its transform, are $\Psi_L^T$ and $S^T$ respectively. Therefore, $\Psi_L$ and $S$ must be full rank and also, the row space of $\Psi_L$ must be identical to the column space of $S$.

The first $LM_T$ columns of $\Psi$ consist of the analysis filter coefficients of the second filter or they are zeros. The last $(N_T - 1)M_T$ columns of $\Psi$ include the analysis filter of the first filter bank or they are zeros. Thus, the matrix $\Psi_L$ can be written as

$$\Psi_L = \begin{bmatrix} \Psi_{2,L} & \Psi_{1,L} \end{bmatrix}$$ (24)

where $\Psi_{2,L}$ are the columns containing only the second analysis filters and $\Psi_{1,L}$ is the submatrix which contains only the first analysis filters. Using equations (56),(19),(21) and Table (1), $\Psi_{1,L}$ and $\Psi_{2,L}$, after eliminating zero columns, can be described as

$$\Psi_{1,L} = \begin{bmatrix} \begin{bmatrix} P_1^T \\ O_{M_1} \\ O_{M_1} \\ \vdots \\ O_{M_1} \end{bmatrix} & \begin{bmatrix} O_{M_1} \\ P_1^T \\ O_{M_1} \\ \vdots \\ O_{M_1} \end{bmatrix} & \cdots & \begin{bmatrix} O_{M_1} \\ \vdots \\ O_{M_1} \end{bmatrix} \end{bmatrix}$$ (25)

$$\Psi_{2,L} = \begin{bmatrix} \begin{bmatrix} P_2^T \\ O_{M_2} \\ O_{M_2} \\ \vdots \\ O_{M_2} \end{bmatrix} & \begin{bmatrix} O_{M_2} \\ P_2^T \\ O_{M_2} \\ \vdots \\ O_{M_2} \end{bmatrix} & \cdots & \begin{bmatrix} O_{M_2} \\ \vdots \\ O_{M_2} \end{bmatrix} \end{bmatrix}$$ (26)
where

\[ M_z = \begin{cases} M_2 & \text{if } M_1 \geq M_2 \\ M_2 - M_1 & \text{otherwise} \end{cases} \]  \hspace{1cm} (27)

\( P_1 \) is defined as (44) when all entries are from the first analysis filter bank. \( P_2 \) is defined the same as \( P_1 \) but all entries are from the second analysis filter bank.

If both filter banks have the same number of bands \((M_1 = M_2 = M)\), the same length of filters \((N_1 = N_2 = N)\) and delay of \(\Delta = N - 1\), then equations (25) and (26) can be written as

\[
\Psi_{1,L} = \begin{bmatrix}
P_{1,0}^T & O_{M_1} & O_{M_1} \\
P_{1,1}^T & P_{1,0}^T & O_{M_1} \\
\vdots & \vdots & \ddots & \vdots \\
P_{1,L-2}^T & P_{1,L-3}^T & O_{M_1} \\
P_{1,L-1}^T & P_{1,L-2}^T & P_{1,0}^T
\end{bmatrix}
\] \hspace{1cm} (28)

\[
\Psi_{2,L} = \begin{bmatrix}
P_{2,L}^T & P_{2,L-1}^T & P_{2,1}^T \\
O_{M_2} & P_{2,L}^T & P_{2,2}^T \\
\vdots & O_{M_2} & \ddots & \vdots \\
O_{M_2} & \vdots & P_{2,L-1}^T \\
O_{M_2} & O_{M_2} & P_{2,L}^T
\end{bmatrix}
\] \hspace{1cm} (29)

where

\[
P_i = \begin{bmatrix}
P_{i,0} \hspace{1cm} P_{i,1} \hspace{1cm} P_{i,2} \hspace{1cm} \cdots \hspace{1cm} P_{i,L}
\end{bmatrix}.
\] \hspace{1cm} (30)

\( P_{i,j} \) is an \(M_i \times M_i\) matrix for \(i = 1, 2\) and \(j = 0, 1, \cdots, L\). The zero columns in equations (28) and (29) are eliminated. Note that \(\Psi_{1,L}\) is the upper part of time-invariant \(A\) matrix [20] for the first filter bank and \(\Psi_{2,L}\) is the lower part of same matrix for the second filter bank. It is clear that the sum of the ranks of the upper part and lower part of matrix \(A\), in a perfect reconstruction system is equal to \(N - M\). The above result shows that to switch one filter bank to another, a similar property should be valid, i.e, the upper part of the first \(A\) matrix and the lower part of the second \(A\) matrix should span the space of \(R^L = R^{N-M}\) together. It can be shown that for \(M = 2\), \(\Psi_{1,L}\) and \(\Psi_{2,L}\) in this case are always half-rank. So to guarantee existence of a proper post filter, it is only necessary that \(\Psi_{1,L}\) and \(\Psi_{2,L}\) span orthogonal subspaces.

**Example 3.1:** Consider the switching from one 2-band system to another 2-band system, both with 8-tap analysis and synthesis filters and a delay of 7 samples. The length of the transition period
is 6 samples. Using (22), $\Pi$ can be written as (after eliminating zero columns):

$$
\begin{bmatrix}
\vdots & P_{2,0} & P_{2,1} & P_{2,2} & P_{2,3} \\
\vdots & P_{2,0} & P_{2,1} & P_{2,2} & P_{2,3} \\
\vdots & P_{2,0} & P_{2,1} & P_{2,2} & P_{2,3} \\
\vdots & P_{1,0} & P_{1,1} & P_{1,2} & P_{1,3} \\
\vdots & P_{1,0} & P_{1,1} & P_{1,2} & P_{1,3} \\
\vdots & P_{1,0} & P_{1,1} & P_{1,2} & P_{1,3} \\
\vdots & P_{1,0} & P_{1,1} & P_{1,2} & P_{1,3} \\
\vdots & P_{1,0} & P_{1,1} & P_{1,2} & P_{1,3} \\
\end{bmatrix}
$$

$\Psi_L$ is shown in the above equation. It is easy to see that $\Psi_L$ is applied to the input data in the transition period. Also,

$$
\Psi_{1,L} = \begin{bmatrix} P_{1,0}^T & O & O \\ P_{1,1}^T & O & O \\ P_{1,2}^T & P_{1,1}^T & P_{1,0}^T \end{bmatrix} \quad \Psi_{2,L} = \begin{bmatrix} P_{2,3}^T & P_{2,2}^T & P_{2,1}^T \\ O & P_{2,3}^T & P_{2,2}^T \\ O & O & P_{2,3}^T \end{bmatrix}
$$

are half rank matrices. So it is enough that they span orthogonal subspaces of $\mathbb{R}^6$.

In summary, $Z_t$ is the overall transform of the time-varying filter bank applied to the input samples during the transition period. In order to obtain the original signal, $Z_t$ must be full rank which means no information is lost in the distorted samples. This condition implies nothing more than the basic requirement for an invertible transform.

### 3.3 The Least Squares and Pseudo Least Squares Synthesis Filters

We have shown that the necessary condition for existence of the post filter is $\det(Z_t) \neq 0$. Although this is a necessary and sufficient condition, it does not guarantee the accuracy of the numerical calculation. From a practical point of view, $Z_t$ should be a well-conditioned matrix.

The analysis sections of the first and second filter banks are defined by the required properties of the first and second filter bank transforms and the only way to improve the condition of $Z_t$ is to choose proper synthesis filters during transition period. In this section, we describe two methods to improve the condition of $Z_t$.

**The Pseudo Least Squares (PLS) synthesis filters:** In this solution, we choose synthesis filters $s(n)$ such that $Z_t$ becomes the identity matrix. Equation (57) can be written as
\[
\begin{bmatrix}
A_a(n) \\
A_t(n) \\
A_b(n)
\end{bmatrix}
\begin{bmatrix}
s(n)
\end{bmatrix}
=
\begin{bmatrix}
O \\
b_t(n)
\end{bmatrix}.
\] (33)

Splitting the above equation into two, we obtain:

\[A_t(n) \ s(n) = b_t(n),\] (34)

\[A_{ab}(n) \ s(n) = O\] (35)

where \(b_t(n)\) is a vector of length \(L\) in which the \((n_0 + L - n)\)th element is 1 and the rest are zero and \(A_{ab}(n) = [A_a(n)^T, A_b(n)^T]^T\). Equation (34) is an underconstrained system and many different \(s(n)\)'s might satisfy it. To reduce distortion, we choose the best answer that satisfies (34) and minimizes the difference of the left and right side of equation (35). The 'best' \(s(n)\) in \(L_2\) norm sense is

\[s(n) = \text{Proj}_{(\alpha(n))}(O)\] (36)

where \(\alpha(n)\) is the subspace spanned by \(A_t(n)\) and \(\text{Proj}_{\alpha}(\cdot)\) is the projection operator on subspace \(\alpha\). This solution not only provides that \(Z_t\) be the identity matrix, but also minimizes distortion of the transition period with the above constraint. In the PLS solution, distortion of the filter bank output during the transition period is generated only from the samples outside of the transition period.

**The Least Squares (LS) synthesis filters:** In this solution, we find the best solution for equation (57) in the least squares sense. As this equation is overconstrained, the best solution is obtained as

\[s(n) = (A(n)^T A(n))^{-1} A(n)^T b(n).\] (37)

This solution minimizes distortion during the transition period. Since \(A(n)s(n)\) is the closest impulse response to \(b(n)\), \(Z_t\) is close to the indentity matrix. Therefore, it usually results in a very well-conditioned \(Z_t\).

**3.4 Comments on Using Near PR Filter Banks**

Applying the post filter to transition periods is based on this assumption that the samples before and after the transition periods are exactly reconstructed. But in many applications, the time-invariant filter banks are not PR but *near perfect reconstruction (NPR)*. The most popular NPR filter banks
are QMF filter banks [24]. The post filter approach may also be applied to NPR filter banks, as discussed next.

To begin, we notice that equations (9) and (10) are obtained when the first and second filter bank are assumed PR. Therefore, the columns of \( W \) corresponding to the samples before and after the transition period, are filled with identity matrices for the ideal impulse responses. If the first and the second filter banks are NPR, those columns are substituted with the corresponding impulse responses \( Z_1 \) and \( Z_2 \):

\[
\hat{x}_K(n_0 + L + \Delta - 1) = \hat{W}^T x_\hat{K}(n_0 + L + \Delta - 1)
\]

\[
\hat{W} = \begin{bmatrix}
Z_2 \\
Z
\end{bmatrix}
\begin{bmatrix}
Z_1
\end{bmatrix}.
\]

\( Z_1 \) is the impulse response matrix of the filter bank for the time interval \( (n_0 - I + \Delta, n_0) \) and \( Z_2 \) is the impulse response matrix for the time interval \( [n_0 + L, n_0 + L + \Delta) \). If \( Z_1 \) has \( I_1 \) columns, then \( \hat{K} = I + I_1 + L - 2 \). To obtain the input/output relationship of the overall system, we apply equation (38) to (14):

\[
\hat{x}_L(n_0 + L + \Delta + \Theta - 1) = Y^T \hat{W}^T x_\hat{K}(n_0 + L + \Delta - 1). \tag{40}
\]

\( \hat{W}Y \) defines the impulse response of the overall system (including the filter bank and post filter) in the transition period. The distance between \( \hat{W}Y \) and the ideal impulse response matrix is the impulse response error vector

\[
e_T = \begin{bmatrix}
e(n_0) \\
e(n_0 + 1) \\
e(n_0 + 2) \\
\vdots \\
e(n_0 + L - 1)
\end{bmatrix} = \left\| \hat{W}Y - \begin{bmatrix}
O_{2\Delta} \\
I_L \\
O_{I+I_1-2\Delta-2}
\end{bmatrix} \right\|_T \tag{41}
\]

and the ASNR of the transition period is equal to

\[
\text{ASNR} = -10 \log_{10}(\frac{1}{L}e_T^T e_T). \tag{42}
\]

4 Design Examples

In this section, we present three examples of time-varying filter bank design using post filtering. In all examples, the time-varying filter bank consists of two time-invariant filter banks, switchable to each
other. Such a time-varying filter bank has 2 different transition periods: the transition associated with switching the first filter bank to the second filter bank (denoted as the forward transition) and the transition period associated with switching back from the second filter bank to the first one (denoted as the backward transition). We use filter banks that are independently designed to be PR or NPR. Five different scenarios for switching synthesis filters are considered:

1. Direct switching of the synthesis filters at the beginning of the transition period.
2. Direct switching of the synthesis filters at the end of the transition period.
3. Direct switching of the synthesis filters at the moment where the ASNR is maximum over all switching delay times.
5. Switching using LS synthesis filters.

For each switching scenario, the filter bank and post filter impulse responses are calculated. In order to compare the reconstruction quality, the ASNR matrices of the filter bank and overall system are calculated for each switching method.

**Example 4.1, Dual 2-band 8-tap System:** The first example is a time-varying filter bank consisting of two 2-band systems with 8-tap analysis and synthesis filters with system delay of 7-samples. These filter banks are independently designed to be very closed to PR (i.e ASNRs are about 160 dB). The analysis filters' frequency characteristics are shown in Figure (7). The ASNR matrices at the outputs of the filter bank and the post filter are included in Table (2). The last row of this table shows the time-varying filter bank which is obtained by optimization for forward switching [10]. The condition numbers of $Z_i$'s are also shown in this table.

**Example 4.2, Dual 2-band 12-tap QMF System:** This example demonstrates the behavior of time-varying filter banks consisting of NPR time-invariant filter banks. The 12-tap type A and B Johnston QMFs [1] are used in this example. The ASNRs for these filter banks is about 50 dB. In this case, the length of each transition period is 10 samples. The ASNR matrices at the outputs of the filter bank, the post filter and the condition numbers of the $Z_i$'s are shown in Table (3).
Figure 7: Frequency characteristics of two independent 2-band perfect reconstructing filter banks used in Example 4.1. Full lines show the first set and dotted lines show the second set.

<table>
<thead>
<tr>
<th>Filter Bank ASNR matrix (dB)</th>
<th>Post Filter ASNR matrix (dB)</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct at the beginning</td>
<td>167.6 2.8 160.0</td>
<td>167.6 175.7 168.9 160.0</td>
</tr>
<tr>
<td>Direct at the end</td>
<td>167.6 2.8 160.0</td>
<td>167.6 167.4 175.7 160.0</td>
</tr>
<tr>
<td>Best Direct Switching</td>
<td>167.6 10.5 160.0</td>
<td>167.6 194.3 194.4 160.0</td>
</tr>
<tr>
<td>PLS</td>
<td>167.6 29.3 28.8 160.0</td>
<td>167.6 203.3 204.0 160.0</td>
</tr>
<tr>
<td>LS</td>
<td>167.6 29.3 28.9 160.0</td>
<td>167.6 203.4 204.7 160.0</td>
</tr>
<tr>
<td>NPR</td>
<td>97.0 101.8 51.5 141.5</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 2: The average signal-to-reconstruction noise ratio (ASNR) of the time-varying filter bank of Example 4.1, at the output of the filter bank and post filter for different switching methods of the synthesis section. The condition vectors include the condition numbers of forward (top) and backward (bottom) switchings.
Table 3: The average signal-to-reconstruction noise ratio (ASNR) of the time-varying filter bank of Example 4.2, at the output of the filter bank and post filter for different switching methods of the synthesis section. The condition vectors include the condition numbers of forward (top) and backward (bottom) switchings.

Example 4.3, 2-band/3-band System: In this example, a time-varying filter bank with dual frequency resolution is designed. We use the 2-band and 3-band filter banks of Section 2.2. The ASNR matrices at the outputs of the filter bank, the post filter and the condition numbers of the $Z_t$'s are listed in Table (4). The last row of Table (4) shows the time-varying filter bank which is obtained by optimization [10].

As is seen in Tables (2), (3) and (4), ASNRs are significantly improved after post filtering in all cases. Also note that $Z_t$ is very well-conditioned in PLS and LS design methods compared to direct switching.

5 Conclusion

Perfect reconstructing time-varying filter banks were discussed in this paper and a novel design method based on time-varying post filtering was presented. The time-varying filter bank was obtained by switching between different time-invariant perfect reconstructing filter banks. It was shown that
Table 4: The average signal-to-reconstruction noise ratio (ASNR) of the time-varying filter bank of Example 4.3, at the output of the filter bank and post filter for different switching methods of the synthesis section. The condition vectors include the condition numbers of forward (top) and backward (bottom) switchings.
direct switching between the time-invariant filter banks usually results in substantial distortion during transition periods. It was also shown that the time-varying filter bank can be considered as a block transform around the transition period. Therefore, an inverse block transform can be used to eliminate distortion in the transition period. The time-varying post filter is the implementation form of this inverse transform. The necessary and sufficient condition for post filtering are described in terms of the analysis/synthesis sections of the time-invariant filter banks. Practical issues such as the effect of imperfections of the time-invariant filter banks and numerical accuracy of calculations have also been addressed.

It was shown that time-varying post filtering not only provides instantaneous analysis transform switching but also exact reconstruction of the input signal at all times. Due to the simplicity of the design procedure, it can be easily applied in real time systems. Since the post filter approach treats the general case of linear analysis/synthesis filter banks, it can obviously be applied to more restricted subsets such as octave-band systems (or discrete wavelets as they are now being called) and an infinite variety of hierarchal tree structures. The theory may also be extended to the multidimensional case [19].

A Time-Domain Formulation For Time-Varying Filter Banks

The time-domain formulation for time-invariant filter banks was presented by Nayebi et al. [20]. In [10], the general time-domain formulation of time-varying filter banks is reported. In this appendix, we present a new derivation for the time-domain formulation.

Figure (8) shows the diagram of a time-varying filter bank. In this figure, the filter bank is divided into three stages: the analysis filters, the down/up samplers and the synthesis filters. The signals \( x(n) \) and \( \hat{x}(n) \) are the filter bank input and output at time \( n \) respectively. The outputs of the analysis filters are shown by \( v(n) = [v_0(n), v_1(n), \ldots, v_{M(n)-1}(n)]^T \), where \( v_i(n) \) is the output of the \( i^{th} \) analysis filter at time \( n \). The outputs of the down/up samplers at time \( n \) is called \( w(n) = [w_0(n), w_1(n), \ldots, w_{M(n)-1}(n)]^T \).

The input/output relation of the analysis filters can be expressed by

\[
v(n) = P(n)x_N(n) .
\] (43)
Figure 8: Time-varying filter bank as cascade of analysis filters, down/up samplers and synthesis filters.

$P(n)$ is an $M(n) \times N(n)$ matrix whose $m^{th}$ row is comprised of the coefficients of the $m^{th}$ analysis filter at time $n$:

$$P(n) = \begin{bmatrix}
    h_0(n,0) & h_0(n,1) & h_0(n,2) & \ldots & h_0(n,N(n) - 1) \\
    h_1(n,0) & h_1(n,1) & h_1(n,2) & \ldots & h_1(n,N(n) - 1) \\
    h_2(n,0) & h_2(n,1) & h_2(n,2) & \ldots & h_2(n,N(n) - 1) \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    h_{M(n)-1}(n,0) & h_{M(n)-1}(n,1) & h_{M(n)-1}(n,2) & \ldots & h_{M(n)-1}(n,N(n) - 1) 
\end{bmatrix}$$

(44)

where $h_i(n,j)$ is the $j^{th}$ coefficient of the $i^{th}$ analysis filter and $x_N(n)$ is the input vector of length $N(n)$ at time $n$:

$$x_N(n) = [x(n), x(n-1), x(n-2), \ldots, x(n - N(n) + 1)]^T.$$  

(45)

The input/output function of down/up samplers can be expressed in the form

$$w(n) = \Lambda(n)v(n)$$

(46)

where $\Lambda(n)$ is a diagonal matrix of size $M(n) \times M(n)$. The $m^{th}$ diagonal element of $\Lambda(n)$, at time $n$, is 1 if the input and output of the $m^{th}$ down/up sampler are identical, otherwise it is zero.

To write the input/output relationship of the synthesis filters, $Q(n)$ is defined as

$$Q(n) = \begin{bmatrix}
    g_0(n,0) & g_0(n,1) & g_0(n,2) & \ldots & g_0(n,N(n) - 1) \\
    g_1(n,0) & g_1(n,1) & g_1(n,2) & \ldots & g_1(n,N(n) - 1) \\
    g_2(n,0) & g_2(n,1) & g_2(n,2) & \ldots & g_2(n,N(n) - 1) \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    g_{M(n)-1}(n,0) & g_{M(n)-1}(n,1) & g_{M(n)-1}(n,2) & \ldots & g_{M(n)-1}(n,N(n) - 1) 
\end{bmatrix}$$

$$= \begin{bmatrix} q_0(n) & q_1(n) & q_2(n) & \ldots & q_{N(n)-1}(n) \end{bmatrix}$$

(47)

where $q_i(n) = [g_0(n,i), g_1(n,i), g_2(n,i), \ldots, g_{M(n)-1}(n,i)]^T$, is a vector of length $M(n)$ and $g_i(n,j)$ denotes the $j^{th}$ coefficient of the $i^{th}$ synthesis filter. At time $n$, the $m^{th}$ synthesis filter is convolved
with vector \([w_m(n), w_m(n-1), \ldots, w_m(n - N(n) + 1)]^T\) and all outputs are added together. Using (47), the output of the filter bank at time \(n\) can be written as:

\[
\hat{x}(n) = \sum_{i=0}^{N(n)-1} q_i^T(n) w(n - i).
\]

If \(s(n)\) and \(\hat{w}(n)\) are defined as

\[
s(n) = [q_0^T(n), q_1^T(n), q_2^T, \ldots, q_{N(n)-1}^T(n)]^T
\]

\[
\hat{w}(n) = [w^T(n), w^T(n - 1), w^T(n - 2), \ldots, w^T(n - N(n) + 1)]^T,
\]

then equation (48) can be written in the form of one inner product,

\[
\hat{x}(n) = s^T(n)\hat{w}(n)
\]

where \(s(n)\) and \(\hat{w}(n)\) are vectors of length \(N(n)M(n)\). Using equations (51),(50),(46) and (43), the input/output function of the filter bank can be written as:

\[
\hat{x}(n) = s^T(n) \\
\begin{bmatrix}
A(n) P(n) x_N(n) \\
A(n-1) P(n-1) x_N(n-1) \\
A(n-2) P(n-2) x_N(n-2) \\
\vdots \\
A(n - N(n) + 1) P(n - N(n) + 1) x_N(n - N(n) + 1)
\end{bmatrix}
\]

As the last \(N(n) - 1\) elements of vector \(x_N(n - i)\) are identical to the first \(N(n) - 1\) elements of vector \(x_N(n - i - 1)\), the latter equation can be expressed by

\[
\hat{x}(n) = s^T(n) \\
\begin{bmatrix}
A(n) P(n) & \cdots & O \\
O & A(n-1) P(n-1) & \cdots & O \\
O & O & A(n-2) P(n-2) & \cdots & O \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
O & \cdots & O & A(n - N(n) + 1) P(n - N(n) + 1) & \cdots & O
\end{bmatrix}
\begin{bmatrix}
x(n) \\
x(n-1) \\
x(n-2) \\
\vdots \\
x(n - 2N(n) + 1)
\end{bmatrix}
\]

where \(O\) is the zero column vector with length \(M(n)\). Thus, the input/output function of a time-varying filter bank can be expressed in the form of

\[
\hat{x}(n) = z^T(n)x_I(n)
\]
where \(x_1(n) = [x(n), x(n-1), \ldots, x(n-I+1)]^T\) and \(I(n) = 2N(n) - 1\) and \(z(n)\) is the time-varying impulse response vector of the filter bank at time \(n\):

\[
z(n) = A(n) s(n).
\]  (55)

The matrix \(A(n)\) is the \([2N(n) - 1] \times [N(n) M(n)]\) matrix

\[
A(n) = \begin{bmatrix}
P(n)^T A(n) & O^T & & \\
O^T & P(n-1)^T A(n-1) & O^T & \\
& & \ddots & O^T \\
& & O^T & P(n-N(n)+1)^T A(n-N(n)+1)
\end{bmatrix}.
\]  (56)

For a perfect reconstruction filter bank with a delay of \(\Delta\), it is necessary and sufficient that all elements but the \((\Delta + 1)^{th}\) in \(z(n)\) be equal to zero at all times. The \((\Delta + 1)^{th}\) entry of \(z(n)\) must be equal to one. If the ideal impulse response is \(b(n)\), the filter bank is PR if and only if

\[
A(n) s(n) = b(n) \quad \text{for all } n.
\]  (57)

References


Necessary Conditions for the Optimality of Variable Rate Residual Vector Quantizers*

Faouzi Kossentini and Mark J. T. Smith
Digital Signal Processing Laboratory
School of Electrical Engineering
Georgia Institute of Technology
Atlanta, GA 30332

Christopher F. Barnes
Georgia Tech Research Institute
Georgia Institute of Technology
Atlanta, GA 30332

June 15, 1993

Abstract—Residual vector quantization (RVQ), or multistage VQ, as it is also called, has recently been shown to be a competitive technique for data compression [1]. The competitive performance of RVQ reported in [1] results from the joint optimization of variable rate encoding and RVQ direct-sum codebooks. In this paper, necessary conditions for the optimality of variable rate RVQs are derived, and an iterative descent algorithm based on a Lagrangian formulation is introduced for designing RVQs having minimum average distortion subject to an entropy constraint. Simulation results for these entropy-constrained RVQs (EC-RVQs) are presented for memoryless Gaussian, Laplacian, and uniform sources. A Gauss-Markov source is also considered. The performance is superior to that of entropy-constrained scalar quantizers (EC-SQs) and practical entropy-constrained vector quantizers (EC-VQs), and is competitive with that of some of the best source coding techniques that have appeared in the literature.

Index Terms—Residual vector quantization, multistage vector quantization, entropy, source coding.

*This work was supported in part by the National Science Foundation under contract MIP-9116113 and by NASA.
1 Introduction

Residual Vector Quantization (RVQ), or multistage VQ, as it is also called, was originally introduced in 1982 [2]. Its structure, which is shown in Figure 1, consists of a cascade of VQ stages (hence the name multistage VQ). For the pth stage VQ the input vector \( z_p \) is quantized resulting in the approximation \( \hat{z}_p \). The difference is then computed to form the residual \( z_{p+1} = z_p - \hat{z}_p \), which serves as an input to the next stage. This aspect of the structure motivates the name residual VQ or RVQ.

Perhaps the most striking benefit of RVQ is its memory efficient structure. An RVQ with \( P \) stages and \( N_p \) vectors per stage \((1 \leq p \leq P)\) can uniquely represent \( \prod_{p=1}^{P} N_p \) vectors with only \( \sum_{p=1}^{P} N_p \) vectors needed for storage. Furthermore, similar savings in computation may be achieved by exploiting the RVQ tree structure.

Despite these attractive features, RVQ has received little attention until recently. Early assessments of its utility, as reported by Baker [3] and in a survey paper by Makhoul, et al. [4], were somewhat discouraging. In the former case, some preliminary investigations with RVQ structures having more than two stages (applied to image coding) led to the conclusion that it is not advantageous to iteratively vector quantize image waveform residuals [3, p. 102]. In the latter case, Makhoul, et al. observed a rapid degradation in performance for RVQ applied to speech coding as the number of stages was increased and suggested that RVQ be limited to not more than two or three stages.

In 1989, Barnes [5] introduced an analysis of RVQ in which the RVQ is optimized subject to the imposed structural constraint. The new design method led to an improvement in performance over previous design methods. Since then the technical literature has shown much activity in the area of RVQ and the application of RVQ to data compression has become more widespread [6, 7, 8, 9, 10, 11, 12, 13].
In this paper, we extend the theory and design methods of fixed rate RVQs to the case of variable rate RVQ. The first part of the paper (Section 2) follows the work presented in [14, 15] where necessary conditions for the optimality of fixed rate RVQ are derived. Here, however, we present a mathematical treatment of convergence for the RVQ design algorithm. The next part of the paper gives a derivation of optimality conditions for variable rate RVQ. It is well known that variable rate systems can yield a lower average rate than fixed rate systems. This property has been demonstrated in [16, 17] for entropy-constrained VQ (EC-VQ). EC-VQ has shown some of the best performance results among entropy coded quantization schemes. In our discussions, a theory for entropy constrained RVQ (EC-RVQ) is developed. In addition, a locally optimal design algorithm is introduced and convergence issues are addressed. The paper concludes with an evaluation and comparison of the performance of EC-RVQ on some well-known synthetic sources. Simulation results show that EC-RVQ achieves some of the best performance results reported to date.

2 Fixed Rate RVQ

The first approach introduced for the design of RVQs consists of using the LBG algorithm sequentially on each stage [2]. Although each of the stage codebooks is designed to minimize the average distortion introduced by that stage (given fixed prior stages), there is no guarantee that the overall average distortion introduced by the RVQ is minimized. A better design technique is one that designs the stage codebooks jointly to minimize the overall average distortion. The key to optimizing the RVQ stages jointly is to view the RVQ in terms of a structurally constrained direct-sum codebook (that is, a codebook that contains all possible ordered direct-sums of stage code vectors) and find necessary conditions for the optimality of that
direct-sum codebook (i.e., joint optimality of all stage codebooks).

A direct-sum codebook may be depicted in several ways. Here we choose to view it diagramatically as a tree. To illustrate this, consider a three-stage RVQ with two vectors in each stage codebook: stage 1 contains vectors \( y_1(1), y_1(2) \); stage 2 contains vectors \( y_2(1), y_2(2) \); and stage 3 contains \( y_3(1), y_3(2) \). Figure 2 shows a tree corresponding to this RVQ where the stages are delineated by the dashed lines and the stage code vectors appear inside the nodes. Eight nodes appear at the base of the tree, each one corresponding to a direct-sum code vector. The value of any one of the eight code vectors is obtained by tracing the unique path from bottom to top and summing the stage code vectors (shown inside the nodes) along the way. This simple tree interpretation is helpful for suggesting efficient RVQ encoder structures, and for understanding both the optimality conditions and the corresponding RVQ design algorithms.

Equally important to the discussion is the mathematical notation used to describe inputs, outputs, and the various components of the RVQ. Let \( \mathbf{x}_1 \) be a realization of the random \( k \)-dimensional vector \( \mathbf{X}_1 \) described by the probability density function (pdf) \( f_{\mathbf{X}_1}(\mathbf{x}_1) \) on \( \mathbb{R}^k \). A \( P \)-stage RVQ (see Figure 1) consists of a finite sequence of \( P \) vector quantizers. For the \( p \)th stage VQ where \( 1 \leq p \leq P \), let us define the following symbols:

- \( N_p \): the \( p \)th stage codebook size
- \( j_p \): the \( p \)th stage index: \( \{1 \leq j_p \leq N_p\} \)
- \( J_p \): the \( p \)th set of all possible values for \( j_p \): i.e. \( \{1, 2, \ldots, N_p\} \)
- \( y_{p}(j_p) \): the \( j_p \)th code vector of the \( p \)th stage
- \( S_p(j_p) \): the \( j_p \)th partition cell of the \( p \)th stage
- \( V_p(j_p) \): the \( j_p \)th stage-removed residual equivalent class of the \( p \)th stage
- \( C_p \): the \( p \)th stage codebook \( \{y_{p}(j_p) : j_p \in J_p\} \)
- \( P_p \): the \( p \)th stage partition \( \{S_p(j_p) : j_p \in J_p\} \)
- \( Q_p \): the \( p \)th stage quantizer mapping
The $p$th stage VQ quantizes the residual vector $x_p$ and outputs $Q_p(x_p)$. The $p$th stage quantizer mapping $Q_p : \mathbb{R}^k \rightarrow C_p$ can be realized by a composition of a fixed length encoder mapping $E_p : \mathbb{R}^k \rightarrow J_p$ where

$$E_p(x_p) = j_p \text{ if and only if } x_p \in S_p(j_p),$$

and a fixed length decoder mapping $D_p : J_p \rightarrow C_p$ where

$$D_p(j_p) = y_p(j_p).$$

As stated in the previous section, a $P$-stage RVQ can be represented by a tree as illustrated in Figure 2. The associated "single-stage" direct-sum VQ codebook and the tree-structured RVQ codebook are identical in the sense that they produce the same representation of the source output, and thus, have the same expected distortion. For the direct-sum VQ, let us define the following symbols:

- $N$: direct-sum codebook size ($N = \prod_{p=1}^{P} N_p$)
- $J$: direct-sum $P$-tuple index set, $J = J_1 \times J_2 \times \cdots \times J_P$
- $j$: a $P$-tuple index in $J$
- $y(j)$: $j$th direct-sum code vector
- $V(j)$: $j$th direct-sum partition cell
- $C$: direct-sum codebook $\{y(j) : j \in J\}$
- $P$: direct-sum partition $\{V(j) : j \in J\}$
- $Q$: direct-sum mapping

The direct-sum codebook contains all possible ordered sums of the stage code vectors, i.e., $C = C_1 + C_2 + \ldots + C_P$. The direct-sum code vectors are given by

$$y(j) = \sum_{p=1}^{P} y_p(j_p),$$

where $j_p$ is the $p$th member of the ordered $P$-tuple index $j$. The direct-sum VQ quantizes the source vector $x_1$ and outputs the representation $\hat{x}_1 = Q(x_1)$ given by

$$Q(x_1) = \sum_{p=1}^{P} Q_p(x_p),$$
where
\[ x_p = x_1 - \sum_{i=1}^{p-1} Q_i(x_i), \quad p > 1, \]
is the \( p \)th stage causal residual. The term causal refers to the sequential process used to compute the residual, i.e., the stage residuals are computed sequentially starting from the first stage to the \( p \)th stage.

2.1 Necessary Conditions for Optimal Fixed Rate RVQ

Let the distortion that results from representing \( x \) with \( y \) be expressed by \( d(x, y) \). The distortion measure \( d(x, y) \) is assumed to be a non-negative real-valued function that satisfies the following requirements:

1. For any fixed \( x \in \mathbb{R}^k \), \( d(x, y) \) is a continuously differentiable function of \( y \in \mathbb{R}^k \).
2. \( d(x, y) \) is translation invariant.
3. For any fixed \( x \in \mathbb{R}^k \), \( d(x, y) \) is a strictly convex function of \( y \), that is, \( \forall y_1, y_2 \in \mathbb{R}^k \) and \( \lambda \in (0, 1) \),
   \[ d(x, \lambda y_1 + (1 - \lambda)y_2) < \lambda d(x, y_1) + (1 - \lambda)d(x, y_2). \]

A \( P \)-stage RVQ is said to be optimal if it gives at least a locally minimum value of the average distortion incurred in representing \( x_1 \) with \( \hat{x}_1 \),

\[ D(x_1, \hat{x}_1) = E \left\{ d \left[ x_1, \sum_{p=1}^{P} Q_p(x_p) \right] \right\}. \tag{1} \]

For stage codebook and partition optimality, (1) should be minimized with respect to stage codebook and partition parameters. However, this minimization is complicated by the fact that knowledge of the joint pdf \( f_{X_1 \ldots X_P}(x_1, \ldots, x_P) \) is required, which, in turn, depends in a complicated fashion upon the sequence of stage codebooks and partitions. This optimization problem can be made tractable by viewing the RVQ
product code as a single-stage VQ with a structurally constrained direct-sum codebook (i.e., the direct-sum code vectors are structurally dependent). By minimizing the average distortion of the direct-sum quantizer,

\[ D(x_1, \hat{x}_1) = E\{d(x_1, Q(x_1))\}, \]

the problem of dealing explicitly with the complicated structural interdependencies that exist among the stages of the RVQ is avoided.

First, to derive optimality conditions for a fixed rate RVQ direct-sum partition, assume that the stage codebooks \( \{C_1, C_2, \ldots, C_p\} \) are fixed, which implies that the direct-sum codebook \( C \) is also fixed. Then

\[ E\{d[x_1, Q(x_1)]\} \geq E\left\{ \min_{y(j) \in C} d[x_1, y(j)] \right\}. \]

That is, no direct-sum partition can yield lower average distortion than the partition obtained by the nearest-neighbor mapping. Accordingly, we have the nearest-neighbor encoding rule,

\[ x_1 \in V^*(j) \iff d[x_1, y(j)] \leq d[x_1, y(k)] \text{ for all } k \in J. \quad (2) \]

The optimal direct-sum partition cells are denoted with asterisks, \( V^*(j) \).

The next step is to determine necessary conditions for optimal stage code vectors. For the derivation that follows it is useful to introduce the *stage-removed* index mapping \( \beta_p : J \mapsto \tilde{J}_p, \tilde{J}_p = J_1 \times J_2 \times \cdots \times J_{p-1} \times J_{p+1} \times \cdots \times J_p, \) defined by

\[ \beta_p(j) = (j_1, j_2, \ldots, j_{p-1}, j_{p+1}, \ldots, j_p) \]

for \( j \in J \). Note that \( \beta_p(j) \) includes all members of \( j \) except the \( p \)th member, hence the name *stage-removed* index. This index represents a shortened path through the RVQ tree where the \( p \)th level branch has been removed, and the remainder of the path starting with the \((p+1)\)th level branch has been added or
grafted back into the tree structure. Hence, each direct-sum code vector \( y(j) \), where 
\( j = (j_1, j_2, \ldots, j_{p-1}, j_p, j_{p+1}, \ldots, j_P) \in J \), can be written as

\[
y(j) = g(\beta_p(j)) + v_p(j_p),
\]

where

\[
g(\beta_p(j)) = \sum_{i=1, i \neq p}^{P} v_i(j_i)
\]
is the \( p \)th stage-removed direct-sum path of the RVQ tree.

Given a particular \( x_1 \in \mathbb{R}^k \) and a fixed RVQ encoding rule, there exists a \( p \)th stage-removed residual vector defined by

\[
\gamma_p = x_1 - g(\beta_p(j)).
\]

This residual vector is the difference between the input and the stage-removed direct-sum vector. Because the stage-removed residual \( \gamma_p \) is a translation (conditioned on the \( p \)th stage) of the realization \( x_1 \) of the random vector \( X_1 \), it is also a realization of a random vector \( \Gamma_p \) with associated stage-removed residual probability density function \( f_{\Gamma_p}(\gamma_p) \).

In addition, let \( H_p(j_p) \) be the set of \( P \)-tuple indices corresponding to all direct-sum code vectors \( y(j) \) that contain \( y_p(j_p) \) in their construction. In other words, \( H_p(j_p) \subset J \) is the set of all indices such that \( j_p \in J_p \) is the \( p \)th element of \( j \). The set \( H_p(j_p) \) can be used to describe the \( j_p \)th stage-removed residual equivalence class \( V_p(j_p) \) by

\[
V_p(j_p) = \bigcup_{j \in H_p(j_p)} (V(j) - g(\beta_p(j))),
\]

where \( V(j) - g(\beta_p(j)) \) indicates that all \( x_1 \in V(j) \) have been translated by \( g(\beta_p(j)) \).

If \( V(j) \) is assumed to be an optimal partition, i.e. \( V(j) = V^*(j) \), then \( V_p(j_p) = V_p^*(j_p) \) is an optimal stage-removed residual equivalence class.
To determine necessary conditions for optimal stage code vectors, assume that the stage partitions \( \{ \mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_P \} \) are fixed, which implies that the direct-sum partition \( \mathcal{P} \) is fixed. Now let \( K_p \) be the set of all possible \( p \)th stage codebooks \( \mathcal{C}_p \) with \( N_p \) vectors, and let \( \mathcal{K} \) be the set of all possible direct-sum codebooks formed from the \( K_p \)'s with \( 1 \leq p \leq P \). Also let \( F : \mathcal{K} \mapsto [0, \infty) \) be the function given by

\[
F(C) = \sum_{j \in J} E_{X_1} \{ d(x_1, y(j)) | x_1 \in V(j) \} \Pr \{ x_1 \in V(j) \},
\]

for \( y(j) \in C \) and \( C \in \mathcal{K} \). To find a minimum for the average distortion (4), it suffices to find a sequence of codebooks \( (\mathcal{C}_1^*, \mathcal{C}_2^*, \ldots, \mathcal{C}_p^*) \in K_1 \times K_2 \times \ldots \times K_P \) and corresponding direct-sum codebook \( \mathcal{C}_p^* \in \mathcal{K} \) that minimizes \( F \). Coordinate descent algorithms can be used to find such a minimum. These algorithms are based on the following procedure: we hold fixed all stage codebooks, except for the \( p \)th stage codebook, and then we minimize \( F \) with respect to \( \mathcal{C}_p \). This is an iterative procedure and is performed for each stage (i.e. all values of \( p \)) until \( F(C) \) converges to a minimum. There are two common forms of implementation [18]. In the first, often called the nonlinear Jacobi algorithm, the minimizations with respect to the different codebooks \( \{ \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_P \} \) are carried out simultaneously. Mathematically, the nonlinear Jacobi algorithm is described by

\[
\mathcal{C}_p(t + 1) = \arg \min_{\mathcal{C}_p} F(\mathcal{C}_1(t), \ldots, \mathcal{C}_{p-1}(t), \mathcal{C}_p, \mathcal{C}_{p+1}(t), \ldots, \mathcal{C}_P(t)),
\]

for \( 1 \leq p \leq P \). In the second approach, often called the nonlinear Gauss-Seidel algorithm, the minimizations are carried out successively for each codebook and may be described mathematically by

\[
\mathcal{C}_p(t + 1) = \arg \min_{\mathcal{C}_p} F(\mathcal{C}_1(t + 1), \ldots, \mathcal{C}_{p-1}(t + 1), \mathcal{C}_p, \mathcal{C}_{p+1}(t), \ldots, \mathcal{C}_P(t)),
\]
for $1 \leq p \leq P$. Let us assume all stage codebooks (except for the $p$th stage codebook) are fixed. Also, let us modify (4) by writing

$$F(C) = \sum_{j_p \in J_p} \sum_{\beta_p(j) \in J_p} E \{ d(\mathbf{x}_1, \mathbf{y}_p(j)) + \mathbf{y}_p(j_p) \mid \mathbf{x}_1 \in V(j) \} \Pr \{ \mathbf{x}_1 \in V(j) \}. \quad (7)$$

Using the assumption that the distortion measure is translation invariant, and also using (3) together with the law of total probability, we can rewrite the above equation as

$$F(C) = \sum_{j_p \in J_p} E \Gamma_{\rho(j_p)} \left\{ d(\mathbf{y}_p(j_p), \mathbf{y}_p(j_p)) \mid \mathbf{y}_p \in V_p(j_p) \right\} \Pr \{ \mathbf{y}_p \in V_p(j_p) \}$$

$$\geq \sum_{j_p \in J_p} \inf_{\mathbf{u} \in \mathbb{R}^k} E \Gamma_{\rho(j_p)} \left\{ d(\mathbf{y}_p(j), \mathbf{u}) \mid \mathbf{y}_p \in V_p(j_p) \right\} \Pr \{ \mathbf{y}_p \in V_p(j_p) \}. \quad (7)$$

In [19], it is shown that provided $\Pr \{ \mathbf{y}_p \in V_p(j_p) \} \neq 0$, there exist $\mathbf{y}_p^*(j_p) \in \mathbb{R}^k$ (which we call stage-removed residual centroids) for the stage-removed residual equivalence classes $V_p(j_p)$ such that

$$\int d(\mathbf{y}_p(j_p), \mathbf{y}_p^*(j_p)) f_{\Gamma_{\rho(j_p)}}(\mathbf{y}_p(j_p)) d\mathbf{y}_p = \inf_{\mathbf{u} \in \mathbb{R}^k} \int d(\mathbf{y}_p(j), \mathbf{u}) f_{\Gamma_{\rho(j_p)}}(\mathbf{y}_p(j)) d\mathbf{y}_p < \infty, \quad (8)$$

and that the set of all solutions $\mathbf{y}_p^*(j_p)$ to (8) is convex, closed, and bounded. Since the distortion measure $d(x, y)$ is assumed to be strictly convex in $y$, the solution is unique. In (8) the pdf $f_{\Gamma_{\rho(j_p)}}(\mathbf{y}_p(j_p))$ is related to the source pdf $f_{X_1}(\mathbf{x}_1)$ according to

$$f_{\Gamma_{\rho(j_p)}}(\mathbf{y}_p(j_p)) = \sum_{j \in H_{\rho(j_p)}} f[V(j)] f_{X_1} \left[ g(\beta_p(j)) + \mathbf{y}_p(j) \right] \Pr \{ \mathbf{y}_p \in V_p(j_p) \}, \quad (9)$$

where $f[V(j)]$ is an indicator function for the direct-sum partition cell $V(j)$, that is, $f[V(j)] = 1$ if $\mathbf{x}_1 \in V(j)$ and $f[V(j)] = 0$ otherwise. The $\mathbf{y}_p(j_p)$'s which satisfy (8) are generalized centroids of stage-removed residual vectors (i.e., residual vectors formed from the encodings of all prior and subsequent RVQ stages). Hence, the second condition will be referred to as the stage-removed residual centroid condition.
Convergence of the nonlinear Gauss-Seidel algorithm applied to RVQ can now be established using a descent approach.

**Proposition 1:** Suppose $F$ is continuously differentiable and convex on $K_1 \times K_2 \times \ldots \times K_p$. Furthermore, suppose that for each $p \in \{1, 2, \ldots, P\}$, $F$ is a strictly convex function of $C_p$ when the other codebooks are held fixed. Let $\{(C_1(t), \ldots, C_P(t))\}$ with $t = 0, 1, 2, \ldots$ be a sequence of stage codebooks generated by the nonlinear Gauss-Seidel algorithm. Then, every limit point of $\{(C_1(t), \ldots, C_P(t))\}$ minimizes $F$ over $K_1 \times K_2 \times \ldots \times K_p$.

Details of the convergence proof are given in [20]. The proof is based on a descent approach. In particular, successive minimizations cannot increase the value of $F[C_1(t), \ldots, C_P(t)]$. This shows that $F[C_1(t+1), \ldots, C_P(t+1)] \leq F[C_1(t), \ldots, C_P(t)]$ and implies the convergence of $F[C_1(t), \ldots, C_P(t)]$ provided that $F$ is bounded below. It should be noted that if $F$ is not differentiable, the Gauss-Seidel algorithm may fail to converge to a minimum.

The proof outlined above does not apply to the Jacobi algorithm. Even though minimizations with respect to each stage cannot increase the value of $F$, the fact that these minimizations are carried out simultaneously allows the possibility that $F[C_1(t+1), \ldots, C_P(t+1)] > F[C_1(t), \ldots, C_P(t)]$. However, convergence of the nonlinear Jacobi algorithm can be established under suitable assumptions on the new codebook selection rule or mapping $R: K_1 \times K_2 \times \ldots \times K_p \rightarrow K_1 \times K_2 \times \ldots \times K_p$, given by

$$R(C_1, C_2, \ldots, C_P) = (C_1, C_2, \ldots, C_P) - c\nabla F(C_1, C_2, \ldots, C_P),$$

(10)

where $c$ is a positive real number and $\nabla F$ denotes the gradient of $F$ [21].

**Proposition 2:** Let $F$ be a continuously differentiable function, let $c$ be a real number, and suppose that the mapping $R(C_1, C_2, \ldots, C_P)$ given by (10) is a contraction mapping with respect to the block-max norm $B(C_1, C_2, \ldots, C_P) = \max_p ||C_p||_p / w_p$, ...
where each $|| \cdot ||_p$ is the Euclidean norm on $K_p$ and each $w_p$ is a positive real number. Then, there exists a unique vector $(C_1^*, C_2^*, \ldots, C_P^*)$ that minimizes $F$ over $K_1 \times K_2 \times \ldots \times K_P$. Moreover, the sequence $\{(C_1(t), \ldots, C_P(t))\}$ generated by either of the two algorithms (described by (5) and (6)) converges to $(C_1^*, C_2^*, \ldots, C_P^*)$ geometrically. For proof, see [20].

A common distortion measure is the squared error distortion measure defined by

$$d(x, y) = ||x - y||^2 = \sum_{i=1}^{k} (x_i - y_i)^2,$$

where $|| \cdot ||$ denotes the Euclidean norm and $x_i$ and $y_i$ are elements of the vectors $x$ and $y$, respectively. This distortion measure can be written in the form

$$d(x, y) = \rho(||x - y||)$$

where $\rho(\alpha) = \alpha^2$. Obviously, $\rho$ is a continuously differentiable and strictly convex function on $[0, \infty)$ with $\rho(0) = 0$. It follows that the squared error distortion measure satisfies the requirements (1)-(3) in Section 2.1. Therefore, it can be easily shown that $F$ is continuously differentiable and convex on $K_1 \times K_2 \times \ldots \times K_P$, and that $F$ is a strictly convex function of $C_p$. Thus, Proposition 1 guarantees that when the squared error distortion is used, the nonlinear Gauss-Seidel algorithm converges to a minimum.

A necessary condition for $R(C_1, C_2, \ldots, C_P)$ to be a contraction mapping is that $x - c[\rho(x)]'$ be a contraction mapping for any positive real number $c$. It is clear that the function $\rho(x) = x^2$ does not satisfy such a requirement, and Proposition 2 cannot be used to guarantee the convergence of the Jacobi algorithm (when the squared error distortion measure is used). In fact, computer simulations confirm the Jacobi algorithm is not guaranteed to converge, even when the initial vector is close to $(C_1^*, C_2^*, \ldots, C_P^*)$. 

12
2.2 The RVQ Design Algorithm

The RVQ design algorithm, introduced in [5], attempts to jointly optimize all stage codebooks to minimize the overall reconstruction error of the RVQ subject to a constraint on the number of direct-sum code vectors. It is an iterative procedure that is similar to the LBG algorithm. However, unlike the LBG algorithm, the optimization of the decoder (assuming the encoder is fixed) is an embedded iterative procedure that guarantees that new stage codebooks minimize the overall average distortion introduced by the RVQ. Therefore, there are two interlaced iterative procedures: one for optimization of the encoder/decoder pair, and another to simultaneously satisfy the stage-removed residual centroid condition in all stages.

Assuming that all stage codebooks are held fixed, the first optimality condition (given by (2)) implies that only exhaustive search encoders are guaranteed, in general, to generate an optimal direct-sum Voronoi partition. However, exhaustive search encoding is usually too expensive. An alternative (but generally sub-optimal) encoder is the stage-sequential encoder. Although fast, this encoder is often unable to find the best direct-sum code vector, thereby resulting in what may be a significant increase in average distortion. Another sub-optimal, but efficient and effective encoder is the $M$-search encoder. The $M$-search technique, introduced in [22] for tree searching, was shown to be very efficient when used to search the RVQ tree [23, 15]. The $M$-search algorithm proceeds one level deeper into the RVQ tree by extending all branches from $M$ surviving nodes, and only the best $M$ of these extended branches survive to the next level. This procedure continues until the last stage of the codebook is reached, and then the code vector of the best path among the final $M$ paths is used. Employing $M$-search during the optimization of the encoder usually leads to a relatively small complexity, but to close-to-optimal performance [23, 15].
Assuming a fixed direct-sum partition $P$, or equivalently, a fixed set of stage partitions $\{P_1, P_2, \ldots, P_P\}$, the Gauss-Seidel algorithm is used to find the constituent codebooks $\{C_1^*, C_2^*, \ldots, C_P^*\}$ with stage code vectors that simultaneously satisfy the stage-removed residual centroid condition (8). It is shown above that, for the squared error distortion measure, the Gauss-Seidel algorithm always converges to a minimum. Therefore, the "decoder-only" iteration used to find a minimizing set of stage codebooks can only reduce or leave unchanged the average distortion.

It is shown in [20] that if the encoder yields a Voronoi partition (in the squared error distortion sense) with respect to the direct-sum codebook and the Gauss-Seidel algorithm is used in the decoder optimization step, the fixed rate RVQ design algorithm converges monotonically to a fixed point which satisfies necessary conditions for minimum squared error distortion. However, it should be emphasized that if a sub-optimal encoder is used, then the encoder optimization step may actually increase the average distortion and monotonic convergence cannot be guaranteed. The possibility of a nonmonotonic average squared error distortion raises the issue of how to effectively terminate the iterative process. Fortunately, experimental results show that the stage-sequential search RVQ design algorithm effectively reduces the average distortion with only occasional deviations from monotonicity. Furthermore, the $M$-search RVQ design algorithm converged monotonically in all our experiments to a fixed point.

3 Variable Rate RVQ

An optimal variable rate RVQ can be constructed by incorporating the entropy constraint directly into the RVQ design loop. In [1], it is shown that the direct-sum codebook constraint can generally be expected to lead to both an increased average
distortion and a decreased output entropy. This motivates an RVQ design algorithm which finds stage code vectors that minimize the average distortion subject to a constraint on the output entropy of the RVQ. Necessary conditions for optimality of variable rate RVQ are derived in the next section, and an entropy-constrained RVQ design algorithm which satisfies these conditions is discussed in the following section.

3.1 Necessary Conditions for Optimal Variable Rate RVQ

For the direct-sum VQ, let $\mathcal{J}$ be set of variable length indices \{c(j) : j ∈ $\mathcal{J}$\}. The direct-sum VQ mapping, $Q : \mathbb{R}^k → C$, may be realized by a composition of a variable length encoder mapping $E : \mathbb{R}^k → \mathcal{J}$, where

$$E(x_1) = c(j) \text{ if and only if } x_1 ∈ V(j),$$

and a variable length decoder mapping $D : \mathcal{J} → C$ where

$$D(c(j)) = y(j).$$

The variable length encoder can be further decomposed into two mappings, $E = L \circ E$, where $E : \mathbb{R}^k → J$ and $L : J → \mathcal{J}$, and $\circ$ denotes composition. Similarly, one can decompose the variable length decoder into two mappings, $D = D \circ (L)^{-1}$, where $(L)^{-1} : \mathcal{J} → J$, and $D : J → C$. Note that the mapping $L$ is one-to-one and onto, and hence, is an invertible mapping with inverse $(L)^{-1}$.

Let the distortion that results from representing $x_1$ with $\hat{x}_1$, $d(x_1, \hat{x}_1)$, be a non-negative real-valued function that satisfies requirements (1)-(3) of Section 2.1. According to distortion-rate theory [24],[25], [26], the $k$th-order distortion function (where $k$ is the vector size)

$$D_k(R) = \inf_{pr\{\hat{x}_1 | x_1\}} \{E[d(x_1, \hat{x}_1)] \mid I(x_1; \hat{x}_1) ≤ R\}$$
is a lower bound to the \( k \)th-order operational distortion-rate function

\[
\hat{D}_k(R) = \inf_{(\mathcal{E}, \mathcal{D})} \{ E[d(x_1, \hat{x}_1)] | E[l(x_1)] \leq R \}
\]

where \( l(x_1) = |\mathcal{E}(x_1)| \) is the length of the codeword representing \( x_1 \) and \( I(x_1; \hat{x}_1) \) is the mutual information between \( x_1 \) and \( \hat{x}_1 \). The convex hull of \( \hat{D}_k(R) \) can be found [16] by minimizing the functional

\[
J(\mathcal{E}, \mathcal{D}) = E[d(x_1, \hat{x}_1)] + \lambda E[l(x_1)]
\]

where \( \lambda \) can be interpreted as the slope of a line supporting the convex hull of the operational distortion-rate function \( \hat{D}_k(R) \).

A variable rate \( P \)-stage RVQ (with an average rate no greater than \( R \)) is said to be optimal for \( f_{X_1}(\cdot) \) if it gives at least a locally minimum value of the average distortion. The design problem can be stated as follows: Choose the codebook \( C \), partition \( P \), and variable-length mapping \( L \) that minimize the average distortion

\[
D(x_1, \hat{x}_1) = E\{d(x_1, Q(x_1))\}
\]

subject to

\[
E\{l(x_1)\} \leq R,
\]

where \( l : \mathbb{R}^k \rightarrow \mathbb{R} \) is the variable length of the codeword representing \( x_1 \), and is defined by

\[
l(x_1) = |\mathcal{E}(x_1)| = |L(E(x_1))| = |L(j)|.
\]

This constrained minimization problem can be replaced by the following unconstrained minimization problem: Choose the codebook \( C \), partition \( P \), and variable length mapping \( L \) that minimize the Lagrangian

\[
J_\lambda(E, L, D) = E\{d(x_1, \hat{x}_1) + \lambda |L(j)|\}. \quad (11)
\]
Proceeding, assume the codebooks \( \{C_1, C_2, \ldots, C_P\} \) are fixed. This implies the direct-sum codebook \( C \) is fixed. Also, assume the lengths \( |L(j)| \) of the channel codewords associated with the direct-sum code vectors are fixed. Then, a partition \( P \) that minimizes (11) is one that minimizes the integrand \( d(x_1, \hat{x}_1) + \lambda |L(j)| \) almost everywhere. That is,

\[
x_1 \in V^*(j) \text{ iff } d[x_1, y(j)] + \lambda |L(j)| \leq d[x_1, y(k)] + \lambda |L(k)| \text{ for all } k \in J.
\]

(12)

Note that (2) is a special case of (12) when \( \lambda = 0 \).

Next, assume the codebooks \( \{C_1, C_2, \ldots, C_P\} \) and the partitions \( \{P_1, P_2, \ldots, P_P\} \) are fixed. This implies that both the direct-sum codebook \( C \) and the direct-sum partition \( P \) are fixed. Then, note that (11) can be expressed as

\[
J_\lambda(E, L, D) = \sum_{j \in J} E \{d[x_1, y(j)] + \lambda |L(j)| \mid x_1 \in V(j)\} \operatorname{pr}(j)
\]

(13)

where \( \operatorname{pr}(j) = \operatorname{pr}\{x_1 \in V(j)\} \). A mapping \( L \) that minimizes (13) is one that minimizes the expected codeword length

\[
R = \sum_{j \in J} |L(j)| \operatorname{pr}(j).
\]

Setting the codeword length \( |L(j)| \) to

\[
|L^*(j)| = -\log_2 \operatorname{pr}(j) = -\log_2 \operatorname{pr}(j_1, j_2, \ldots, j_P)
\]

(14)

results in an average rate which is equal to the output entropy of the direct-sum quantizer.

The probability \( \operatorname{pr}(j_1, j_2, \ldots, j_P) \) of a path in the RVQ can also be written as the product of conditional probabilities, i.e.,

\[
\operatorname{pr}(j_1, j_2, \ldots, j_P) = \operatorname{pr}(j_P \mid j_{P-1}, \ldots, j_1) \operatorname{pr}(j_{P-1} \mid j_{P-2}, \ldots, j_1) \ldots \operatorname{pr}(j_2 \mid j_1) \operatorname{pr}(j_1)
\]
Therefore, we can write

\[ |L^*(j)| = -\log_2 \text{pr}(j_P|j_{P-1}, \ldots, j_1) - \log_2 \text{pr}(j_{P-1}|j_{P-2}, \ldots, j_1) \]

\[ \cdots - \log_2 \text{pr}(j_2|j_1) - \log_2 \text{pr}(j_1) \]  \hspace{1cm} (15)

and the output entropy of the optimal direct-sum RVQ can be written as

\[ H^*(J_1, J_2, \ldots, J_P) = \sum_{p=1}^{P} H(J_p|J_{p-1}, \ldots, J_1). \]

Finally, assume the stage partitions \( \{P_1, P_2, \ldots, P_P\} \) are fixed. This implies the direct-sum partition \( P \) is fixed. Also, assume that the lengths \( |L(j)| \) of the channel codewords associated with the direct-sum code vectors are fixed. Then, rewrite (11) as

\[ J_\lambda(E, L, D) = \sum_{j \in J} E \{d[x_1, D(j)] \mid x_1 \in V(j)\} \text{pr}(j) + \]

\[ \lambda \sum_{j \in J} E \{|L(j)| \mid x_1 \in V(j)\} \text{pr}(j). \]  \hspace{1cm} (16)

Clearly, a mapping \( D \) that minimizes (16) is one that minimizes

\[ \sum_{j \in J} E \{d[x_1, D(j)] \mid x_1 \in V(j)\} \text{pr}(j). \]

To achieve this minimum, the multistage code vectors \( y_p(j_p) \) at the \( p \)th stage must satisfy (8), i.e.,

\[ \int d[\gamma_p, y_p(j_p)] f_{\gamma_p}(\gamma_p) d\gamma_p = \inf_{u \in \mathbb{R}^k} \int d(\gamma_p, u) f_{\gamma_p}(\gamma_p) d\gamma_p, \]  \hspace{1cm} (17)

where \( \gamma_p = x_1 - g(\beta_p(j)) \), and \( f_{\gamma_p}(\gamma_p) \) is defined by (9).

### 3.2 The EC-RVQ Design Algorithm

The EC-RVQ design algorithm proposed here is an iterative descent algorithm similar to the one used for the design of EC-VQ codebooks. Each iteration consists of
applying the transformation

\[(E(t + 1), L(t + 1), D(t + 1)) = T(E(t), L(t), D(t))\]

where

\[E(t + 1) = \arg \min_E (E(t), L(t), D(t)) \quad \text{(optimum partitions)}\]

\[L(t + 1) = \arg \min_L (E(t + 1), L(t), D(t)) \quad \text{(optimum codeword lengths)}\]

\[D(t + 1) = \arg \min_D (E(t + 1), L(t + 1), D) \quad \text{(optimum code vectors)}\]

Following the lines of argument of [27], one can show that every limit point of the sequence \((E(t), L(t), D(t)), t = 0, 1, \ldots\), generated by the transformation \(T\) minimizes the Lagrangian \(J_\lambda(E, L, D)\) (as given by (11)). Therefore, the EC-RVQ design algorithm is guaranteed to converge to a local minimum.

To find several points on the convex hull of the operational rate-distortion curve, the minimization of \(J_\lambda(E, L, D)\) is repeated for various \(\lambda\)'s. Starting with \(\lambda = 0\) (which corresponds to the RVQ codebook designed by the fixed rate RVQ design algorithm), the EC-RVQ design algorithm uses a pre-determined sequence of \(\lambda\)'s to design locally optimal variable rate EC-RVQ codebooks.

For optimal performance, the EC-RVQ design algorithm must generally employ an exhaustive-search encoder, a jointly optimized direct-sum decoder, and an optimal entropy coder as described by (14). Unfortunately, the computational complexity and memory requirements associated with optimal EC-RVQs are usually prohibitive, and sub-optimal design procedures are usually used to generate practical EC-RVQs.

As with fixed rate RVQ design algorithms, the encoder does not necessarily have to be optimal to be useful. Sub-optimal tree-structured searching techniques such as stage-sequential searching or multipath searching can be employed, leading to

19
relatively fast encoder implementations. Experimental results indicate that stage-sequential searching usually leads to a significant increase in average distortion, but multipath $M$-searching can result in a close-to-optimal performance, even with values of $M$ as small as 2 or 3 [23, 15].

Ideally, all stage codebooks in the RVQ should be jointly optimized. However, since the complexity of the joint optimization design process increases rapidly (quadratically) with increasing number of stages, the RVQ design effort can become excessive. The complexity of the design can be greatly reduced by using conventional stage-sequential optimization, but the resulting performance can also be significantly reduced. The performance gap between sequential and joint optimization can be bridged by local joint optimization of the stage codebooks. The optimization is local in the sense that the stages are partitioned into overlapping blocks and the joint optimization process is restricted to only the stages of each block. This technique was previously employed to accelerate the design of large-block fixed rate RVQ codebooks with a relatively large number of stages [28]. However, we also note that, unlike fixed rate RVQ, EC-RVQ (with a modest number of stages) is shown experimentally to generally perform quite well when sequential stage-wise optimization is used. This encouraging result implies that, at moderate bit rates, the EC-RVQ design speed can be substantially increased without significantly impairing performance.

A unique complexity reducing feature of EC-RVQ is its potential to use stage-conditional (i.e., conditioned on previous stages) entropy tables of relatively small sizes. Equation (15) shows that the optimal length (given by (14)) of the variable length codeword associated with an index $j \in J$ is also the sum of $P$ stage-conditional self-information components. During the design process, the lengths of the stage-conditional entropy codewords can be estimated by using a sufficiently large training
set. Clearly, the aggregate number of tables of stage-conditional entropy codes can become extremely large as the number of stages increases, which may offset the memory savings obtained by using the RVQ structure. However, the number of tables can be made relatively small by limiting the number $m$ of previous stages upon which conditioning is based. In other words, the direct-sum codeword length $|L(j)|$ is approximated by

$$|L(j)|_m = -\log_2 \text{pr}(j_p | j_{p-1}, \ldots, j_{p-m}) - \log_2 \text{pr}(j_{p-1} | j_{p-2}, \ldots, j_{p-m})$$

$$- \ldots - \log_2 \text{pr}(j_2 | j_1) - \log_2 \text{pr}(j_1).$$

(18)

Obviously, since $H(J_p | J_{p-1}, \ldots, J_1) \leq H(J_{p} | J_{p-1}, \ldots, J_{p-m})$ for each $p = 1, 2, \ldots, P$ and $m < p-1$, it is easy to show that $H_m(J) = \sum_{p=1}^{P} H(J_p | J_{p-1}, \ldots, J_{p-m}) \geq H(J)$. Experimental results show that the value of $m$ that results in a good complexity/performance tradeoff increases with both increasing number of stages and vector size, but decreases with increasing stage codebook size. Recent results also show that the best value for $m$ depends heavily on the source. For sources with memory, the best value of $m$ is usually small ($0 < m < 2$). For memoryless sources, however, a larger value of $m$ is usually needed for a good tradeoff, which results in increased memory requirements.

While the sub-optimal EC-RVQ design algorithms discussed above are not guaranteed to converge to local minima, they provide good complexity/performance trade-offs, and they facilitate the design of practical EC-RVQs. We also point out that the sub-optimal algorithms employed in all EC-RVQ experiments performed in this work converged monotonically to a fixed point, and occasional deviations from monotonicity were observed only when stage-sequential searching was used during the encoding step of the EC-RVQ design.
4 Experimental Results

Many quantization techniques have been used to code Gaussian, Laplacian, and uniform memoryless sources, as well as Gauss-Markov sources. Table 1 shows some of the well-known coders as compared qualitatively with EC-RVQ in terms of encoder complexity and memory. For the class of VQ-based coders, EC-RVQ is less demanding in terms of both memory and encoding complexity. It has comparable encoding complexity and memory requirements to that of EC-TCQ but does not suffer from the relatively large coding delays associated with large trellises. Finally, it should be noted that when the dimension is one, EC-RVQ, or entropy-constrained residual scalar quantization (EC-RSQ), has the simplest encoding complexity and the smallest memory requirements.

In this paper we report on the relative performance of these coding techniques for memoryless Gaussian, Laplacian, and uniform sources as well as a Gauss-Markov source. Experimental results demonstrate the performance of EC-RVQ and show its advantages and disadvantages when compared to some of the competitive coding techniques that have appeared in the literature. In particular, EC-RVQ performance is compared to that of scalar quantization (SQ), entropy-constrained SQ (EC-SQ), entropy-constrained VQ (EC-VQ), trellis coded quantization (TCQ), entropy-constrained TCQ (EC-TCQ), and lattice-based VQs. For each of the sources considered here, the EC-RVQs, the EC-RSQs, and the EC-SQs, which are described in Table 2, were designed on training sequences rather than on the underlying distributions, and were used to encode a test sequence of 40,000 samples taken from the same source. The performance results for EC-VQ [16], TCQ [29], EC-TCQ, predictive EC-SQ (PEC-SQ), predictive EC-TCQ (PEC-TCQ) [30], and lattice-based VQs [31, 16] are taken from the literature.
Experimental results for a Gaussian random variable with zero mean and unit variance are shown in Figure 3 (top) and Table 3. Figure 3 (top) shows the rate-distortion performance for the various EC-RVQs and EC-SQ relative to the R(D) curve. Signal-to-noise ratio (SNR) values for EC-RVQ, EC-VQ, EC-SQ, D4 lattice, A2 lattice, TCQ, EC-TCQ, and R(D) at 0.5, 1.0, 1.5, and 2.0 bits per sample (bps) are given in Table 3. It can be seen that the performance of EC-RVQ increases with increased vector size, and that practical EC-RVQs outperform practical EC-VQs with the same vector size, even while maintaining relatively small encoding complexity and memory requirements. EC-RVQ is also competitive with both TCQ and EC-TCQ.

The next set of experiments considers the Laplacian source with zero mean and unit variance. Figure 3 (bottom) shows the rate-distortion performance of several EC-RVQs and EC-SQ relative to a curve linearly interpolated from well-known R(D) points. Numerical values are given in Table 4 for EC-RVQ, EC-RSQ, EC-SQ, TCQ, EC-TCQ, SQ, VQ, and R(D) at 0.5, 1.0, and 2.0 bps. Unlike the case of the Gaussian source, increasing the vector size does not improve the EC-RVQ rate-distortion performance significantly. This is explained by the fact that, as the vector size increases, encoding complexity and memory requirements limit the size of the initial codebook (or the peak bit rate) that can be used to design practical EC-RVQs. This leads to a reduction in rate-distortion performance because the Laplacian source (which has a peaked distribution) requires a very large output alphabet size (i.e., number of levels or code vectors), which is difficult to attain in practice. In fact, EC-RSQ is very competitive with EC-RVQ because the former has the potential to use an expanded set of direct-sum code vectors. When compared to other coding techniques, EC-RVQ (including the special case where the vector size $k$ is equal to 1) outperforms the other coders at low bit rates and is competitive with EC-TCQ at high rates.
Simulation results for the encoding of a memoryless uniform source are shown in Figure 4 (top) with numerical values given in Table 5. As stated in [29], entropy coding does not lead to any performance gains in the case of scalar or trellis coded quantization. However, although the source is uniform, RVQ outputs are generally not equiprobable, and entropy coding usually leads to a slight performance gain. As can be seen, increasing the vector size leads to an increase in rate-distortion performance. However, EC-RVQ performance generally falls below that of TCQ [29], but becomes competitive when the vector size is relatively large (e.g., \( k = 16 \)).

Finally, results for a Gauss-Markov source with correlation coefficient \( \rho = 0.9 \) are shown in Figure 4 (bottom) and Table 6. Again, Figure 6 shows the rate-distortion performance of several EC-RVQs and EC-SQ relative to \( R(D) \) while Table 6 shows the SNRs for a number of predictive coding techniques as well as EC-RVQ and EC-VQ at bit rates of 0.5, 1.0, 1.5, 2.0 and 2.5 bps. It should be noted that for rates \( R > 0.926 \), the \( R(D) \) curve in Figure 4 (bottom) is actually an upper bound on the true derived \( R(D) \) curve. As expected, there is a clear advantage of VQ-based coders over most of the other non-predictive scalar coders. Although EC-VQ is expected to theoretically outperform all VQ-based coders for such a source, practical EC-VQs do not meet that expectation, mainly because the encoding complexity and memory requirements associated with such coders severely limit the initial codebook size (or the peak bit rate). In fact, EC-VQ is significantly outperformed by EC-RVQ with the same vector size. For vector sizes larger than 6, EC-RVQ outperforms PEC-SQ at all bit rates between 0.5 and 2.0 bits/sample, and is competitive with PEC-TCQ, especially at relatively large vector sizes (e.g., \( k = 16 \)). It should be noted that the memory inherent in both the state and the predictor gives PEC-TCQ an effective vector size which is usually larger than the vector sizes used by the VQ-based coders.
5 Summary

Necessary conditions for optimal variable rate RVQ have been derived, and an iterative descent algorithm for designing locally optimal variable rate EC-RVQ codebooks has been introduced. The RVQ structure is exploited to facilitate the implementation of practical EC-RVQs, which perform well even while maintaining very low encoding complexity and memory requirements.

Experimental results for three memoryless sources and a Gauss-Markov source indicate that practical EC-RVQs have performance advantages over other VQ-based coders, including practical EC-VQs. Although EC-RVQ outperforms TCQ-based coders only at some relatively low bit rates for the Laplacian source, it is usually competitive and has the potential of increased rate-distortion performance when the peak bit rate is increased. Furthermore, encoding complexity and memory requirements of EC-RVQ are comparable to those of TCQ-based coders, but EC-RVQ does not have the disadvantage of the long encoding delays associated with large trellises.
References


List of Figures

1  A $P$-stage residual vector quantizer ........................................ 36
2  A 3-level RVQ tree ..................................................................... 36
3  The $R(D)$ performance of several EC-RVQs and EC-SQ relative to the
   true $R(D)$ curve for the Gaussian (Top) and the Laplacian (Bottom)
   memoryless sources. ................................................................. 37
4  The $R(D)$ performance of several EC-RVQs and EC-SQ relative to the
   true $R(D)$ curve for the uniform source (Top) and the Gauss-Markov
   source (Bottom). ................................................................. 38
List of Tables

1 Qualitative comparison of several entropy-coded quantization systems 32
2 Training set size (TSS) in thousands of vectors, number of stages (NS), stage codebook size (SCS) in vectors, peak bit rate (PBR) in bits/sample, number of search paths (NSP), Markov model order (MMO), number of vector distortion calculations (NVDC) per input vector, codebook memory (CM) in kilobytes, and maximum memory requirements for entropy tables (TM) in kilobytes for EC-RVQ, EC-RSQ, and EC-SQ. 33
3 Performance (SNR in dB) of various source coding schemes for the memoryless Gaussian source at 0.5, 1.0, 1.5, and 2.0 bits per sample. 34
4 Performance (SNR in dB) of various source coding schemes for the memoryless Laplacian source at 0.5, 1.0, and 2.0 bits per sample. 34
5 Performance (SNR in dB) of various source coding schemes for the memoryless uniform source at 0.5, 1.0, and 2.0 and 3.0 bits per sample. 35
6 Performance (SNR in dB) of various source coding schemes for the Gauss-Markov source at 0.5, 1.0, 1.5, 2.0 and 2.5 bits per sample. 35
<table>
<thead>
<tr>
<th>System</th>
<th>Block(Vector) Size</th>
<th>Encoding</th>
<th>Memory</th>
<th>Entropy Coder</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC-SQ</td>
<td>1</td>
<td>simple</td>
<td>very small</td>
<td>very simple</td>
</tr>
<tr>
<td>EC-RSQ</td>
<td>1</td>
<td>very simple</td>
<td>very small</td>
<td>very simple</td>
</tr>
<tr>
<td>A2 Lattice</td>
<td>2</td>
<td>moderate</td>
<td>small</td>
<td>simple</td>
</tr>
<tr>
<td>D4 Lattice</td>
<td>4</td>
<td>moderate</td>
<td>small</td>
<td>simple</td>
</tr>
<tr>
<td>EC-VQ</td>
<td>4</td>
<td>complex</td>
<td>large</td>
<td>complex</td>
</tr>
<tr>
<td>EC-VQ</td>
<td>8</td>
<td>very complex</td>
<td>large</td>
<td>complex</td>
</tr>
<tr>
<td>EC-RVQ</td>
<td>4</td>
<td>simple</td>
<td>small</td>
<td>simple</td>
</tr>
<tr>
<td>EC-RVQ</td>
<td>8</td>
<td>moderate</td>
<td>small</td>
<td>simple</td>
</tr>
<tr>
<td>EC-RVQ</td>
<td>16</td>
<td>complex</td>
<td>moderate</td>
<td>moderate</td>
</tr>
<tr>
<td>EC-TCQ(s=8)</td>
<td>1</td>
<td>simple</td>
<td>small</td>
<td>simple</td>
</tr>
<tr>
<td>EC-TCQ(s=8)</td>
<td>4</td>
<td>moderate</td>
<td>small</td>
<td>moderate</td>
</tr>
<tr>
<td>EC-TCQ(s=128)</td>
<td>1</td>
<td>moderate</td>
<td>moderate</td>
<td>moderate</td>
</tr>
</tbody>
</table>

Table 1: Qualitative comparison of several entropy-coded quantization systems
<table>
<thead>
<tr>
<th></th>
<th>EC-RVQ</th>
<th></th>
<th></th>
<th></th>
<th>EC-RSQ</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>k=4</td>
<td>k=6</td>
<td>k=8</td>
<td>k=12</td>
<td>k=16</td>
<td></td>
</tr>
<tr>
<td>TSS</td>
<td>250</td>
<td>300</td>
<td>400</td>
<td>500</td>
<td>750</td>
<td>200</td>
</tr>
<tr>
<td>NS</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>SCS</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>PBR</td>
<td>4.00</td>
<td>3.33</td>
<td>2.50</td>
<td>2.0</td>
<td>2.0</td>
<td>6.0</td>
</tr>
<tr>
<td>NSP</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>MMO</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>NVDC</td>
<td>128</td>
<td>160</td>
<td>160</td>
<td>288</td>
<td>384</td>
<td>12</td>
</tr>
<tr>
<td>CM</td>
<td>1.02</td>
<td>1.92</td>
<td>2.56</td>
<td>4.61</td>
<td>8.19</td>
<td>0.48</td>
</tr>
<tr>
<td>TM</td>
<td>0.39</td>
<td>6.28</td>
<td>6.28</td>
<td>8.33</td>
<td>12.42</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Table 2: Training set size (TSS) in thousands of vectors, number of stages (NS), stage codebook size (SCS) in vectors, peak bit rate (PBR) in bits/sample, number of search paths (NSP), Markov model order (MMO), number of vector distortion calculations (NVDC) per input vector, codebook memory (CM) in kilobytes, and maximum memory requirements for entropy tables (TM) in kilobytes for EC-RVQ, EC-RSQ, and EC-SQ.
### Table 3: Performance (SNR in dB) of various source coding schemes for the memoryless Gaussian source at 0.5, 1.0, 1.5, and 2.0 bits per sample.

<table>
<thead>
<tr>
<th>Rate</th>
<th>EC-RVQ k=4</th>
<th>EC-VQ k=12</th>
<th>EC-SQ k=4</th>
<th>D4</th>
<th>A2</th>
<th>TCQ s=256</th>
<th>EC-TCQ s=128</th>
<th>R(D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>2.21</td>
<td>2.50</td>
<td>2.20</td>
<td>2.09</td>
<td>2.05</td>
<td>2.17</td>
<td>N/A</td>
<td>3.01</td>
</tr>
<tr>
<td>1.0</td>
<td>5.10</td>
<td>5.38</td>
<td>4.80</td>
<td>4.64</td>
<td>4.55</td>
<td>4.78</td>
<td>5.56</td>
<td>6.02</td>
</tr>
<tr>
<td>1.5</td>
<td>7.80</td>
<td>8.21</td>
<td>7.70</td>
<td>7.57</td>
<td>6.95</td>
<td>7.60</td>
<td>N/A</td>
<td>8.79</td>
</tr>
<tr>
<td>2.0</td>
<td>10.68</td>
<td>N/A</td>
<td>N/A</td>
<td>10.55</td>
<td>N/A</td>
<td>N/A</td>
<td>11.04</td>
<td>12.04</td>
</tr>
</tbody>
</table>

### Table 4: Performance (SNR in dB) of various source coding schemes for the memoryless Laplacian source at 0.5, 1.0, and 2.0 bits per sample.

<table>
<thead>
<tr>
<th>Rate</th>
<th>EC-RVQ k=4</th>
<th>EC-RSQ k=6</th>
<th>EC-SQ k=4</th>
<th>TCQ s=256</th>
<th>EC-TCQ s=128</th>
<th>SQ k=6</th>
<th>VQ k=6</th>
<th>R(D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>3.23</td>
<td>3.27</td>
<td>3.15</td>
<td>3.14</td>
<td>2.20</td>
<td>N/A</td>
<td>N/A</td>
<td>1.97</td>
</tr>
<tr>
<td>1.0</td>
<td>5.91</td>
<td>5.92</td>
<td>5.90</td>
<td>5.79</td>
<td>5.54</td>
<td>4.82</td>
<td>3.01</td>
<td>4.96</td>
</tr>
<tr>
<td>2.0</td>
<td>11.38</td>
<td>11.58</td>
<td>11.50</td>
<td>11.31</td>
<td>11.22</td>
<td>12.35</td>
<td>7.54</td>
<td>12.66</td>
</tr>
</tbody>
</table>
Table 5: Performance (SNR in dB) of various source coding schemes for the memoryless uniform source at 0.5, 1.0, and 2.0 and 3.0 bits per sample.

<table>
<thead>
<tr>
<th>Rate</th>
<th>EC-RVQ k=4</th>
<th>EC-RVQ k=16</th>
<th>EC-SQ s=4</th>
<th>EC-SQ s=256</th>
<th>TCQ s=4</th>
<th>TCQ s=256</th>
<th>SQ N/A</th>
<th>R(D)</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>3.12</td>
<td>3.20</td>
<td>3.08</td>
<td>2.84</td>
<td>3.24</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>6.27</td>
<td>6.39</td>
<td>6.04</td>
<td>6.22</td>
<td>6.58</td>
<td>6.02</td>
<td>6.79</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.0</td>
<td>12.27</td>
<td>12.79</td>
<td>12.08</td>
<td>12.62</td>
<td>13.00</td>
<td>12.04</td>
<td>13.21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>18.58</td>
<td>N/A</td>
<td>18.10</td>
<td>18.83</td>
<td>19.23</td>
<td>18.06</td>
<td>19.42</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Performance (SNR in dB) of various source coding schemes for the Gauss-Markov source at 0.5, 1.0, 1.5, 2.0 and 2.5 bits per sample.

<table>
<thead>
<tr>
<th>Rate</th>
<th>EC-RVQ k=4</th>
<th>EC-RVQ k=6</th>
<th>EC-RVQ k=16</th>
<th>EC-VQ k=4</th>
<th>EC-VQ k=8</th>
<th>PEC-SQ s=8</th>
<th>PEC-TCQ s=8</th>
<th>R(D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>7.45</td>
<td>8.43</td>
<td>9.32</td>
<td>7.10</td>
<td>8.15</td>
<td>N/A</td>
<td>N/A</td>
<td>10.22</td>
</tr>
<tr>
<td>1.0</td>
<td>10.64</td>
<td>11.58</td>
<td>12.36</td>
<td>10.40</td>
<td>11.15</td>
<td>N/A</td>
<td>N/A</td>
<td>13.23</td>
</tr>
<tr>
<td>1.5</td>
<td>13.38</td>
<td>14.29</td>
<td>15.29</td>
<td>12.15</td>
<td>N/A</td>
<td>13.86</td>
<td>15.30</td>
<td>16.26</td>
</tr>
<tr>
<td>2.0</td>
<td>16.15</td>
<td>17.23</td>
<td>N/A</td>
<td>15.80</td>
<td>N/A</td>
<td>17.22</td>
<td>18.38</td>
<td>19.25</td>
</tr>
<tr>
<td>2.5</td>
<td>19.14</td>
<td>20.13</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>20.48</td>
<td>21.41</td>
<td>22.26</td>
</tr>
</tbody>
</table>
Figure 1: A $P$-stage residual vector quantizer

Figure 2: A 3-level RVQ tree
Figure 3: The R(D) performance of several EC-RVQs and EC-SQ relative to the true R(D) curve for the Gaussian (Top) and the Laplacian (Bottom) memoryless sources.
Figure 4: The R(D) performance of several EC-RVQs and EC-SQ relative to the true R(D) curve for the uniform source (Top) and the Gauss-Markov source (Bottom).
COMPRESSION OF EARTH SCIENCE IMAGES USING ADAPTIVE ENTROPY-CONSTRAINED RESIDUAL VECTOR QUANTIZATION

Faouzi Kossentini and Mark J. T. Smith

School of Electrical Engineering
Georgia Institute of Technology
Atlanta, GA 30332

Abstract

This paper introduces some improved techniques for coding earth science images. The fundamental coding method upon which this work is based is, entropy-constrained residual vector quantization or EC-RVQ, is a relatively new method that was shown to be capable of excellent rate-distortion performance and to be robust when applied to images outside the training set used in the design. This paper investigates the incorporation of adaptivity into the EC-RVQ framework. In particular, the conditional entropy tables in the EC-RVQ are adapted to the specific input image being coded. Forward adaptation and backward adaptation are considered in this context and are compared in terms of performance and complexity. Experimental results show that the use of adaptivity (as defined) in conjunction with EC-RVQ leads to a noticeable improvement in coding quality while introducing a relatively small increase in computational complexity.

1. Introduction

Entropy-constrained residual vector quantization (EC-RVQ) is a new technique for data compression that was recently shown to be attractive for coding images. This technique, the essence of which is summarized in the next section, is based on a structured vector quantizer in conjunction with tables of variable length entropy codewords. The entropy tables are used to map the indices to statistically optimized variable length codewords, resulting in an average bit rate close to the first-order output entropy of the residual vector quantization (RVQ) codebook. EC-RVQ has many features that make it attractive for coding satellite and earth science imagery. First, the coding quality is very high. Conventional vector quantization in isolation typically does not perform well in comparison to other mainstream image coding methods such as JPEG. However, EC-RVQ does because, unlike conventional VQ, it also jointly optimizes the code vectors and their associated variable length entropy codewords. In all tests performed thus far, EC-RVQ has significantly outperformed JPEG (in terms of both subjective and objective quality) and is among the most competitive techniques presently reported. Second, it is memory efficient. Vector quantizers rely on codebooks which must be stored in memory and therefore represent a storage cost. The multistage residual structure of EC-RVQ, as discussed in the next section, avoids this problem and typically utilizes orders of magnitude less memory than comparable conventional VQs. Third, the memory efficient nature of EC-RVQ allows for the use of vectors with relatively large sizes. It can be shown that as the vector size increases, the VQ system performance can approach the optimal rate-distortion performance bound.

Fourth, EC-RVQ has an extremely fast decoding capability. The decoding procedure consists only of table look-ups, comparisons, and a summation of a small number of vectors. Thus, decoding is virtually instantaneous. Fifth, EC-RVQ has a reasonable encoding complexity. Even with software realizations, the M-search method employed in the EC-RVQ makes its encoding complexity comparable to that of other techniques in its performance class. More important, however, is that encoding can be made extremely fast if parallelism is exploited. This does come at the expense of increased cost in hardware. Six, EC-RVQ is a natural candidate for 8-bit fixed point arithmetic. This makes it very suitable for implementation with low cost off-the-self fixed-point DSP microprocessors. Finally, we mention that EC-RVQ can support progressive transmission. For narrow bandwidth channels where transmission may be slow, the images can be viewed progressively as bits are being received. Indeed, there are many useful features and benefits of EC-RVQ. A detailed discussion of EC-RVQ can be found in the referenced
The current state of performance of EC-RVQ is by no means an upper bound on what is possible. In this paper, we show how further improvements in quality can be achieved by tracking the local behavior of the image statistics. Here we exploit the local statistical properties of the image in addition to the linear and non-linear spatial dependencies exploited by the EC-RVQ.

Image locality can be exploited by designing a separate codebook for each image or region of an image and sending each codebook as side information\(^2\), or by changing the code vectors during the encoding process\(^3\). In the former case, the improvement in coding quality is somewhat offset by the increase in bit rate due to side information associated with sending the codebooks. In the latter case, the improvement in quality is modest, and may not be worth the extra complexity associated with on-line adaptation.

Two new adaptive schemes are considered in this work. For a given image, the first scheme consists of computing a separate set (or many sets) of tables of conditional entropy codewords based on the input image. These updated tables are then transmitted to the decoder as side information after which the coded image is transmitted. We will call this scheme Forward Adaptive EC-RVQ or FA-EC-RVQ. The second scheme consists of adapting the tables many times (via feedback adaptation) as the image is being coded. We will call this scheme Backward Adaptive EC-RVQ or BA-EC-RVQ.

EC-RVQ has a big advantage here because it can be designed with a relatively small number of small entropy tables. Consequently the data associated with sending the tables typically represents a negligible increase in the bit rate, and the additional complexity in terms of computing new tables is generally small. The following sections provide some background discussion, a presentation of the new FA- and BA-EC-RVQs, and some performance evaluations and comparisons for coded aerial images.

2. Entropy-Constrained RVQ

Residual vector quantization (RVQ) is a structurally constrained variant of VQ that consists of a sequence of VQ stages, each operating on the "residual" of the previous stage. A general block diagram of a \(P\)-stage RVQ is shown in Figure 1. For convenience, the stages can be indexed by \(1 \leq i \leq P\) where the \(i\)th stage VQ contains \(N_i\) code vectors. By examining the structure, it is easy to see that many code vectors can be represented in terms of the direct-sum combination of the \(P\) stage code vectors. In particular, \(N = \prod_{i=1}^{P} N_i\) direct-sum code vectors can be uniquely defined using only \(\sum_{i=1}^{P} N_i\) stage code vectors. This represents a tremendous savings in codebook memory over conventional VQ and often similar savings in computation.

Entropy-constrained residual vector quantization (EC-RVQ) is a residual vector quantization technique that exploits the relatively small output entropy of the RVQ\(^4\). On a simple level, it may be viewed as an RVQ in cascade with a table that maps the fixed-length RVQ codewords to another set of statistically optimized variable length codewords. The design algorithm used to create the EC-RVQ seeks a codebook that minimizes the average distortion subject to a constraint on the output entropy of the RVQ. This is done by iteratively minimizing the Lagrangian

\[
J_\lambda = E[d(x_1, \hat{x}_1)] + \lambda E[l(x_1)],
\]

where

- \(x_1\) is the \(k\)-tuple realization of the input,
- \(\hat{x}_1\) is the \(k\)-tuple realization of the output,
- \(d(x_1, \hat{x}_1)\) is the squared error distortion between the input and output,
- \(l(x_1)\) is the length of the variable-length codeword associated with the input vector, and
- \(\lambda\) has an interpretation as being the slope of a line supporting the convex hull of the operational rate-distortion curve\(^5\).

In the iterative EC-RVQ design algorithm, each iteration tries to simultaneously satisfy the necessary conditions for a fixed real-valued control parameter \(\lambda^2\). Upon convergence each \(\lambda\) corresponds to a point on the R(D) trajectory. Using an appropriately large set of training data, the resulting operational R(D) trajectory that is traced by varying
the control parameter forms a reasonably smooth curve in the sense that the fixed operating points, although finite, are quite dense along the curve. As a result, for a fixed target rate/distortion, one need only try several different values of the parameter and choose the RVQ codebook that produces an average rate/distortion closest to the target rate/distortion.

3. Adaptive EC-RVQ

The entropy tables associated with an EC-RVQ as described thus far are very large and unmanageable for the vector sizes and bit rates targeted in this work. For example, an RVQ at a rate of 1.00 bits/pixel (8 stages, 4 vectors/stage, 4 x 4 vectors) would require $2^{17}$ table entries, which is clearly too much. However, if the entropies are conditioned upon the previous stage or previous few stages, the number of table entries can be made manageable.

Developing this line of reasoning, let $J_p$ be the set of all indices which represent the $p$th stage code vectors and $J$ be the set of all indices which represent all possible direct sum code vectors (i.e., $J = J_1 \times J_2 \times \ldots \times J_p$). Also, let $j_p \in J_p$ be the index of the $p$th stage code vector and $j \in J$ be the index of the direct-sum code vector. Assuming that non-integer codeword lengths can be used, the entropy of a $P$-tuple index (or random variable) $j = (j_1, j_2, \ldots, j_P)$, given by

$$|L^*(j)| = - \log_2 pr(j) = - \log_2 pr(j_1, j_2, \ldots, j_P),$$

is essentially the optimal length of the variable length codeword associated with that index $j$. However, the probability $pr(j_1, j_2, \ldots, j_P)$ of a path in the RVQ can also be written as the product of conditional probabilities, i.e.

$$pr(j_1, j_2, \ldots, j_P) = pr(j_P|j_{P-1}, \ldots, j_1) \cdot pr(j_{P-1}|j_{P-2}, \ldots, j_1) \cdot \ldots \cdot pr(j_2|j_1).$$

Therefore,

$$|L^*(j)| = - \log_2 pr(j_P|j_{P-1}, \ldots, j_1) - \log_2 pr(j_{P-1}|j_{P-2}, \ldots, j_1) - \ldots - \log_2 pr(j_2|j_1) - \log_2 pr(j_1).$$

(1)

Equation (1) shows that such an optimal length is also the sum of $P$ stage conditional entropies. By using a sufficiently large training sequence together with (1), one can find a good estimate of the lengths of the stage codewords. Obviously, the size of the tables increases as the stage codebook size $|J_p|$ increases and the number of the tables increases as both the stage codebook size $|J_p|$ and the number of stages $P$ increase. Therefore, the memory requirements associated with storing the tables can very quickly be unmanageable, even for moderate vector sizes and bit rates. However, by modeling the codewords as a time invariant $m$th-order Markov process, which implies that the conditional probabilities depend only on the last $m$ ($m < p - 1$) stages, or equivalently, the direct-sum codeword length $|L^*(j)|$ is approximated by

$$|L(j)| = - \log_2 pr(j_P|j_{P-1}, \ldots, j_{P-m}) - \log_2 pr(j_{P-1}|j_{P-2}, \ldots, j_{P-m}) - \ldots - \log_2 pr(j_2|j_1) - \log_2 pr(j_1),$$

the memory requirements can be greatly reduced.

As alluded to in the introduction, the use of a small number of tables is key in facilitating the exploitation of image locality. In the following subsection, we consider the first of the adaptive systems.

3.1. Forward Adaptive EC-RVQ (FA-EC-RVQ)

This new FA-EC-RVQ generates a small number of statistically optimized entropy tables, which are then sent to the receiver as side information. Specifically, the input image is first divided into a number of nonoverlapping regions or subimages, from which a different set of tables is generated. Codewords are transmitted to the receiver as before but together with the corresponding tables. Thus, the average bit rate for FA-EC-RVQ contains two components: $r_c$, associated with the transmission of the individual variable length codewords which represent the coded vectors of the images and $r_s$, the side information, related to the transmission of the representative tables for each sub-image. Assuming that the image is divided into $K$ subimages (or regions), the total bit rate is

$$r_t = r_c + Kr_s.$$
Because the number of tables is small, the additional bit rate (or \( K_r \)) is usually not significant when the whole image is used (i.e., \( K = 1 \)), but becomes more significant when \( K \) is increased. The improvement due to having input-based probability tables translates into a substantial reduction in bit rate (by as much as 25%) with about the same quality of performance. For comparison, Figure 2 shows the rate-distortion performance of EC-RVQ and FA-EC-RVQ for the test image MOFFET1 (shown in Figure 4). The EC-RVQ codebooks used to generate Figure 2 contain 8 stages with 16 vectors per stage, leading to a peak bit rate of 2.00 bpp. We observe a significant improvement here over EC-RVQ in terms of rate-distortion performance. Clearly there is a tradeoff in the gain achieved from the use of input specific tables and the cost of having to send the tables as side information. Taking both the quality and side information into account, we observe that the performance gain seems to peak at about \( K = 4 \). Thus for this test image, \( K = 4 \) represents the best tradeoff using FA-EC-RVQ.

Often schemes involving two-pass encoding, like this one, can lead to a substantial increase in computational complexity. Fortunately, such is not the case here. Relatively few distortion calculations per input vector are needed to encode the input image and the number and the size of the entropy tables are small. Hence in the balance, FA-EC-RVQ is an attractive candidate for practical implementation.

3.2. Backward Adaptive EC-RVQ (BA-EC-RVQ)

Image locality can also be exploited by changing the tables of conditional entropy codewords such that the EC-RVQ has the ability to track the slowly varying statistics of many earth science images. This can be done by employing an entropy coding algorithm (such as a Huffman coder) to change the current tables or construct new ones during the encoding stage. The additional complexity associated with real time adaptation is relatively small because only few conditional entropy tables are usually used. We have observed that some improvement in quality can be achieved as shown in Figure 3 (with the same design parameters as in Figure 2) due to this adaptive scheme. Note that, unlike FA-EC-RVQ, this scheme does not require any additional side information because only the coded vectors are used to update the tables of entropy codewords. Therefore, the only additional complexity is the extra time needed to update the entropy coder.

4. Discussion of Simulation Results

In this section, we discuss some of the details related to the comparisons (in terms of both objective and subjective quality) performed in this investigation and show some coding results. In all experiments, the objective performance measure used is the peak signal-to-quantization-noise ratio (PSNR) defined by

\[
\text{PSNR} = -10 \log_{10} \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (x(i,j) - \hat{x}(i,j))^2}{(N^2(255)^2)}
\]

where \( N \times N \) is the size of the image (assumed to be squared) and \( x(i,j) \) and \( \hat{x}(i,j) \) represent the original and coded values (respectively) of the pixel at the \( i \)th row and the \( j \)th column of the image.

A diverse set of 256 \times 256 training images was used to form the training data set. Two aerial test images were selected for comparison: MOFFET1 and MOFFET2. So as not to unfairly bias the experimental results, test images were always excluded.
Fig. 4. The original image MOFFET1.

Fig. 5. The image MOFFET1 coded using JPEG. The bit rate is 0.75 bpp and the PSNR is 28.76 dB.

Fig. 6. The image MOFFET1 coded using BA-EC-RVQ. The bit rate is 0.75 bpp and the PSNR is 30.64 dB.

Fig. 7. The image MOFFET1 coded using FA-EC-RVQ (4 nonoverlapping square sub-images). The bit rate is 0.75 bpp and the PSNR is 31.92 dB.
Fig. 8. The original image MOFFET2

Fig. 10. The image MOFFET2 coded using BA-EC-RVQ. The bit rate is 0.67 bpp and the PSNR is 30.05 dB.

Fig. 9. The image MOFFET2 coded using JPEG. The bit rate is 0.67 bpp and the PSNR is 26.93 dB.

Fig. 11. The image MOFFET2 coded using FA-EC-RVQ (4 nonoverlapping square sub-images). The bit rate is 0.67 bpp and the PSNR is 31.02 dB.
from the training set. In all of the experiments, EC-RVQ codebooks with 10 stages, 16 vectors/per stage and 4 x 4 vectors were used and were designed using the iterative EC-RVQ design algorithm. Here the EC-RVQ was based on a 1st-order Markov assumption, i.e., with m = 1. Since only the previous stage is used to estimate conditional probabilities, the total number of tables is 145: one table for the 1st stage and 16 tables for the remaining 9 stages. The 145 tables implied by the EC-RVQ appear to represent a lot of side information. However, many of these tables are not populated when an image is encoded. Experiments indicate that less than 10% of those tables are populated using one sub-image at a time. Consequently the data rate associated with the side information is relatively low.

Encoding in both the normal operation and in the design of the EC-RVQ is performed using multi-path searching (or M-search). To accelerate the encoding process, a recently introduced fast searching algorithm (based on dynamic M-search) is used, requiring only an average of about 80 vector distortion calculations per input vector. This fast search algorithm achieves quality very close to that of exhaustive search RVQ while maintaining a very low level of computational complexity.

Performance results are consistently high over a wide variety of test images. The figures 4-7 show a composite of four images. Figure 4 shows the original MOFFET1 image. The rest are coded versions of the image using the JPEG standard coding algorithm (Figure 5), the new BA-EC-RVQ (Figure 6), and the new FA-EC-RVQ with 4 nonoverlapping square sub-images (Figure 7). All images in this figure are coded at a rate of 0.75 bpp. As can be seen, the quality is better for FA-EC-RVQ than BA-EC-RVQ. FA-EC-RVQ achieves a PSNR of 31.92 dB while the BA-EC-RVQ results is 30.64 dB. Both are higher than the JPEG standard result which is 28.76 dB.

To illustrate coding performance with another example, the figures 8-11 show the same configuration of images for the MOFFET2 image: the original, JPEG, BA-EC-RVQ, and FA-EC-RVQ. In this example, the coding rate is 0.67 bpp. Again, the quality is better for the adaptive EC-RVQ coders both in terms of appearance and PSNR, and the FA-EC-RVQ has the best performance of those methods tested.

5. Remarks

This investigation of adaptive EC-RVQ indicates that improvements can be achieved over non-adaptive EC-RVQ. The amount of improvement is dependent on the image being coded. In all cases observed, adaptivity led to a noticeable quality improvement. Further improvements should be possible by utilizing alphabet and entropy constrained RVQ (AEC-RVQ), a technique which is presently under study. This also has the potential of reducing the memory storage requirements. Additional gains are expected by using larger vector sizes. At this point, only 4 x 4 vectors have been explored. However, computational and memory demands are manageable with 8 x 8 vectors. Experience with EC-RVQ showed noticeable improvement in going from 4 x 4 to 8 x 8 vector sizes. Similar improvement is anticipated for adaptive EC-RVQ when 8 x 8 vectors are used. Results from future investigations in these directions will be reported in future conferences.

6. References


APPLICATION OF ENTROPY-CONSTRAINED RVQ TO CODING IMAGE SUBBANDS

Faouzi Kossentini, Wilson Chung and Mark J. T. Smith
School of Electrical Engineering
Georgia Institute of Technology
Atlanta, GA 30332

Abstract—This paper reports on a recent study on the application of entropy constrained residual vector quantization (EC-RVQ) to subband image coding. The newly introduced EC-RVQ has produced excellent performance results when applied directly to coding images, and is also more cost efficient than competing VQ methods which have, so far, been reported in the literature. Experimental results show that subband coding with EC-RVQ performs very well and place it among the best techniques presently available in terms of performance. The paper provides a discussion of the new system and presents some performance comparisons.

I. INTRODUCTION

Variable rate coding methods such as Huffman and arithmetic coding are being used extensively in image compression applications. These methods attempt to code at a rate close to the signal entropy and thus tend to produce noticeable improvement over fixed rate image coding methods. Entropy-based coding is used in the JPEG standard and has been examined in the context of vector quantization methods for image compression [2], [3]. A notable example of the latter is the work reported in [11] where alphabet and entropy constrained vector quantization (AEC-VQ) was introduced for coding image subbands. These coding results achieved some of the highest PSNR performance results ever reported in the literature. The improvements seen here are due, in part, to the joint exploitation of vector quantization, which has well-known advantages over scalar quantization, and to variable length entropy coding.

Although the performance of entropy-based VQ coding algorithms has been shown to be excellent [2], [3] [11], there are some practical disadvantages. Two in particular are memory storage requirements and computational complexity. A big step in reducing these problems was made by constraining the source alphabet [11], which led to a substantial reduction in the memory because only one codebook was needed. However, the VQ encoding complexity remains very large.

Entropy-constrained residual vector quantization (EC-RVQ), introduced in [9], has several important advantages over entropy-constrained VQ (EC-VQ) [2], [3] and the AEC-VQ discussed in [11]. In particular, EC-RVQ has been shown to be capable of achieving better quality than EC-VQ (whose performance is slightly better than AEC-VQ). In addition, EC-RVQ has the important advantage of low memory requirements (often a factor of 30 or 40 less than EC-VQ) and reduced computational requirements (only about 5% of that of EC-VQ or AEC-VQ). In light of these attractive features, EC-RVQ was explored as a candidate for encoding image subbands. This paper reports on this investigation and addresses issues related to the EC-RVQ design, subband structure, and subband bit allocation. Many tests were performed in this investigation. Some experimental results are included that illustrate the improvement achievable with the new subband EC-RVQ coding algorithm.

II. ENTROPY-CONSTRAINED RVQ

Residual Vector Quantization (RVQ), often called multistage VQ, is a structurally constrained variant of VQ that has very low memory requirements and can have very low computational complexity. It consists of a sequence of VQ stages, each operating on the “residual” of the previous stage. A general RVQ consisting of P stages (with N_i code vectors in the i-th stage) is capable of uniquely representing \( N = \prod_{i=1}^{P} N_i \) vectors with only \( \sum_{i=1}^{P} N_i \) code vectors required for storage. Thus, the RVQ achieves tremendous savings over conventional VQ in terms of memory requirements, and may achieve similar savings in computations. Necessary conditions for the optimality of fixed rate residual vector quantizers were derived in [1], and a joint-optimization algorithm was used to design fixed rate RVQ codebooks. By using large vector sizes and multipath searching, RVQ was shown to be a very competitive technique for image coding [5].

In [6, 7] various ad-hoc methods were used to design variable rate residual vector quantizers. Some of these
methods exploit the multistage structure of RVQ [6], while others exploit the relatively small output entropy of RVQ codebooks [7]. However, none of these attempts was based on a mathematical formulation that includes entropy explicitly in the VQ design problem. Very recently, necessary conditions for the optimality of variable rate residual vector quantizers were derived [8, 9]. An EC-RVQ design algorithm based on these conditions was also introduced [9], and was used to design variable rate RVQ codebooks for image coding.

The goal of the EC-RVQ algorithm is to seek an RVQ codebook that minimizes the average distortion subject to a constraint on the output entropy of the RVQ. This is done by iteratively minimizing the Lagrangian given by

$$J_\lambda = E[d(z_1, \hat{z}_1)] + \lambda E[l(z_1)],$$

where $z_1$ and $\hat{z}_1$ denote the $k$-tuple realizations of the source and the output respectively, $d(z_1, \hat{z}_1)$ is a distortion measure (assumed here to be the squared error measure), $l(z_1)$ is the length of the variable length codeword associated with $z_1$, and $\lambda$ has an interpretation as the slope of a line supporting the convex hull of the operational rate-distortion curve [2]. Each iteration tries to simultaneously satisfy the necessary conditions given in [9]. Starting with $\lambda = 0$ (which corresponds to the RVQ algorithm), the EC-RVQ algorithm uses a pre-determined sequence of $\lambda$'s to design locally optimal variable rate RVQ codebooks. A more detailed explanation of the algorithm can be found in [8, 9].

Conventional entropy-constrained vector quantizers (EC-VQs) have been shown [3, 2] to outperform fixed rate VQs as well as VQs that are followed by optimal entropy codes. However, the encoding complexity and memory requirements are substantially greater than those of fixed rate VQs. Moreover, the design process can take a very long time (from weeks to months) even when small vector sizes and small codebooks are used. Therefore, the additional quality improvement obtained using the conventional EC-VQ coders may not be worth the extra complexity and memory. In contrast, the encoding complexity, memory requirements, and design complexity of EC-RVQ are relatively small. In addition, EC-RVQ has a substantial advantage in terms of rate-distortion performance, since a significantly larger number of direct sum (sum of stage code vectors) code vectors can be used. Figure 1 shows the PSNR versus rate performance of EC-RVQ and EC-VQ for the Lena image. The vector size is $4 \times 4$. For this example, EC-RVQ is about 30 times more efficient than EC-VQ in both encoding complexity and memory requirements. It is clearly seen from the figure that the new EC-RVQ is superior to the EC-VQ. The peak rate for EC-VQ is limited to be relatively small due to practical computation and memory constraints. However, EC-RVQ is far less restricted in this regard. Taking advantage of this flexibility enables EC-RVQ to achieve better performance than EC-VQ. Similar results were obtained using other vector sizes and other test images.

III. THE SUBBAND CODING STRUCTURE

The theory of subband image coding is well represented in the literature at this point [13], [12], [4]. Hence our discussion is brief and restricted to the some of the specifics that characterize the particular structure investigated here. Among the many factors that must be considered in implementing the analysis/synthesis section of the system is computational efficiency. In this system we employ the efficient IIR allpass polyphase filter banks (shown in Figure 2) as discussed in [4]. The filters used in this system are optimized with respect to both magnitude and step response characteristics so as to provide a fair subjective tradeoff between the aliasing and ringing distortions typically associated with each (respectively).

The subband images, after splitting, are coded with EC-RVQ. Many subband splits could be considered; however, two levels of splitting were used here exclusively. The image was first split into four subbands using the recursive two-band filter bank. The lowest frequency subband image was then split again into four subbands, resulting in a total of seven. Several factors motivate this form of subband decomposition. It has been previously used with success, it is reasonably simple from an implementation viewpoint, and the subband images are suited for VQ coding. To elaborate on the latter point, it is well known that inter-band subband image correlation decreases as the low frequency channel is successively split. Thus the bit rate required to code the low frequency channels becomes large. To avoid having to handle excessively high rate coding of the low frequency subbands with VQ, the number of splits was limited to seven bands—a reason-
This enables the subband coder to employ EC-RVQ for all of the subband images without paying an exorbitant price in terms of storage and complexity.

To achieve good coding performance, bit rates must be carefully allocated among the subbands. The bit rate allocated to each of the $N$ subbands can be determined by minimizing the distortion

$$D = \sum_{i=1}^{N} d_i(r_i)$$

subject to the constraint that $\sum_{i=1}^{N} r_i \leq R$, where $r_i$ is the bit rate of the $i$th subband required to achieve a distortion $d_i(r_i)$, and $R$ is the maximum bit rate for the entire image.

The task at hand is to find the optimal or near-optimal bit allocation. One approach is to find the minimum of equation (2) by exhaustively searching all possibilities. In practice, however, the number of possible subband assignments may be prohibitively large. A more efficient solution, considered here, is based on the algorithm derived in [10] for efficient allocation of bit rates among subbands.

In the procedure, a tree is constructed where the root node has $N$ branches that stem from it, one per subband, and the subtree rooted at the end of each branch $i$ is a unary tree of length $L_i$, where $L_i$ is the number of rate-distortion points available for the $i$th subband. These points, $\{(i,j); 1 \leq j \leq L_i, 1 \leq i \leq N\}$, correspond to the rate-distortion pairs $(b_{i,j},D_{i,j})$ where, for $i = 1, 2, \ldots, N$, $b_{i,1} < b_{i,2} < \ldots < b_{i,L_i}$, and $D_{i,1} \geq D_{i,2} \geq \ldots \geq D_{i,L_i}$. For illustration, a seven-band bit allocation tree is shown in Figure 3.

Let $\mathcal{R}$ be a pruned subtree of the constructed tree $T$, where the subband associated with the $i$-th branch has a bit rate $l_i$ and a distortion $d_i$. The bit rate of the subband/EC-RVQ system associated with $\mathcal{R}$ is $l(\mathcal{R}) = \sum_{i=1}^{N} l_i$ and the distortion is $d(\mathcal{R}) = \sum_{i=1}^{N} d_i$, where $b_{i,1} \leq l_i \leq b_{i,L_i}$ and $D_{i,1} \geq d_i \geq D_{i,L_i}$. The bit rate assignment problem can now be solved by finding the rates $l_1, l_2, \ldots, l_i$ that minimize $d(\mathcal{R})$ subject to $l(\mathcal{R}) \leq R$ ($R$ is the maximum rate) over all pruned subtrees $\mathcal{R} \subseteq T$. The optimal pruning algorithm described in [10] gives such bit rates. Once the optimal bit rate for each subband is determined, the encoder used for a particular subband is the one specified by the node of the branch (associated with that subband) of the optimal tree $\mathcal{R}^*$.

The reconstruction portion of the system consists of recombining the coded subband images via a recursive synthesis filter bank.

IV. EXPERIMENTS AND RESULTS

To evaluate the performance of the subband/EC-RVQ system, several test images were coded and compared to conventional methods. One comparison is given here. Figure 4 shows the original $512 \times 512$ test image Lena (not included in the training set). Figure 5 shows Lena coded with subband/EC-RVQ at 0.1809 bits/pixel (bpp), where the PSNR is 31.05 dB. For comparison, Figure 6 also shows the same image coded with the JPEG standard at 0.1869 bpp, where the PSNR is 29.47 dB. Clearly, the
Figure 4: The original image Lena at 8 bpp

subband EC-RVQ achieves better subjective quality and a higher PSNR. We conjecture that these results can be further improved by using larger vector sizes and higher peak rates in the EC-RVQ design. The results of these experiments will be presented at the conference.

REFERENCES


Figure 5: Lena coded with subband/EC-RVQ at 0.1809 bpp. PSNR is 31.05 dB.

Figure 6: Lena coded with JPEG at 0.1869 bpp. PSNR is 29.47 dB.
Subband Image Coding with Optimal Intra- and Inter-Band Subband Quantization*

Faouzi Kossentini, Wilson C. Chung and Mark J. T. Smith
School of Electrical Engineering
Georgia Institute of Technology
Atlanta, GA 30332

Abstract
Significant progress has been made in the theory and design of analysis/synthesis filter banks and in the independent encoding and decoding subsystems. However, by comparison, little attention has been given to the optimal design of subband quantizers in the sense of minimizing the overall average distortion subject to a constraint on the output rate of the subband system. In this paper, necessary conditions for optimal subband quantization are presented. These conditions can be employed to design quantizers that exploit the linear and non-linear dependencies both within and across the subbands, leading to improvement in performance.

1 Introduction
Unlike transform coding [1, 2, 3], the theory of subband coding [4, 5] has yet to reach a state of maturity. However, significant progress is being made in this direction. For example, Woods and O'Neil [6] show that for many subbands and ideal filters, the subband coding gain is the same as the optimum transform gain. In independent work by Rao and Pearlman [5, 7], the subband coding gain is derived based on an analysis of the first-order entropies in each subband, and assuming ideal filters. Specifically, they show that the gain achieved by subband coding can be attributed to a reduction of the difference between finite order entropy and entropy rate. These results are technically valid only for ideal filters. For non-ideal filters, much of the subband coding gain can still be achieved by using filters with good frequency domain characteristics (i.e., approximately ideal).

While the above theoretical studies address the coding gain that may be achieved by subband coding, they do not address rate-distortion optimality. The problem of optimal subband coding in a rate-distortion sense is discussed by Fisher in [8], where it is shown that for realizable subband coding of a wide sense stationary Gaussian source, with separate encoding of the subband signals, the encoding performance is generally inferior to the rate-distortion function of the source. Fisher also shows that if the Gaussian source is white and/or ideal filters are used, then rate-distortion optimality may be achieved by separately encoding the subbands. However, the analysis in [8] is confined to wide sense stationary Gaussian sources, and does not easily generalize to other non-Gaussian sources with memory.

In this paper, we propose a generalized subband framework that provides a way of determining necessary conditions for optimality of subband quantizers given an arbitrary set of filters and an arbitrary decomposition structure. In particular, it is shown that for a general source with memory, optimal subband coding can be achieved by optimizing the subband encoders, decoders, and entropy coders jointly. Under this new framework, subband quantizers are designed together to minimize the overall average distortion subject to a constraint on the entropy rate of the system. These quantizers exploit both linear and non-linear dependencies that may exist between and within the subbands. While strategies for exploiting inter-band dependencies have been suggested recently in the literature [9], the problem of using arbitrary analysis/synthesis systems in a generalized subband decomposition framework in a rate-distortion optimal way has not been addressed to the best of our knowledge. Another important advantage of this framework is that optimal bit allocation is achieved at each stage of the design process, thereby avoiding another level of processing usually associated with optimal bit allocation algorithms.

In the following discussion, the framework is intro-
duced and the necessary conditions for the optimality of subband quantizers are presented. These conditions are then used as the basis for a locally optimal descent design algorithm which we employ to design jointly optimized subband coders. The paper concludes with some preliminary results that demonstrate the potential of the proposed design algorithm.

2 The Subband Framework

The conventional approach for the design of subband quantizers consists of using various source coding design techniques to independently design optimal or near-optimal quantizers for each subband. Although these techniques minimize the average distortion subject to a constraint on the rate for each subband, there is no guarantee that the overall average distortion (subject to a constraint on the entropy rate of the system) is minimized. As mentioned earlier, overall optimality can be achieved for some Gaussian sources even by separately optimizing each subband quantizer. However, one can always guarantee at least local optimality by designing the subband quantizers collectively. As observed in the case of multistage residual quantization [10, 11], joint optimization can be made easier by viewing the set of subband quantizers in terms of a structurally constrained product quantizer (that is, a quantizer whose output consists of all possible combinations of outputs from the M subband quantizers) and finding necessary conditions for the optimality of that quantizer.

Consider the subband coding system shown in Figures 1 and 2. The input signal \( X \), assumed to be a real-valued stationary source signal, is first decomposed into \( M \) subbands using the analysis transformation \( A = (A_1, \ldots, A_M) \), where \( A_1, \ldots, A_M \) are the \( M \) subband analysis transformations (as shown in the figure). The set of outputs \( X^a = \{X^a_1, \ldots, X^a_M\} \) is encoded using a product fixed-length encoder mapping \( E = (E_1, \ldots, E_M) \). This produces the sequence of product indices \( \{j^n\}^\infty_{n=1} \), where \( j^n \in J \), \( J = J_1 \times \ldots \times J_M \), and \( J_m (1 \leq m \leq M) \) is the set of indices in the \( m \)th band. A variable-length entropy encoder \( L \) is then applied to map the sequence \( \{j^n\}^\infty_{n=1} \) into the sequence of product variable-length indices \( \{c^n\}^\infty_{n=1} \), which are then sent to the channel. The output of the channel is decoded using the inverse mapping \( L^{-1} \) to recover the sequence \( \{j^n\}^\infty_{n=1} \), which is then input to the fixed-length decoder \( D = (D_1, \ldots, D_M) \), resulting in the reconstructed signal \( X^* = \{X^*_1, \ldots, X^*_M\} \). Finally, the synthesis transformation \( S = (S_1, \ldots, S_M) \) takes the

![Figure 1: Generalized Subband Framework (a) Encoder](image1.png)

![Figure 2: Generalized Subband Framework (a) Decoder](image2.png)
reconstructed signal $X^*$ and outputs $\hat{X}$, an approximation of the input signal $X$.

Grouping the transforms and fixed-rate encoders/decoders together as shown in the figure, we can define $F^e = (F_1^e, \ldots, F_M^e)$ and $F^d = (F_1^d, \ldots, F_M^d)$ as the product encoder and decoder non-linear mappings, respectively. Let $F$ be the composition of $F^d$ and $F^e$ (i.e., $F = F^d \circ F^e$). According to [12, 13], the convex hull of the operational distortion-rate function,

$$D(R) = \inf_{(\hat{F}, L)} \left\{ E\left(d(X, F(X))\right) \mid E\left(l\left(L(\hat{F}(X))\right)\right) \leq R \right\},$$

where $l(L(\hat{F}(X)))$ denotes the length of the code-word $L(\hat{F}(X))$ and $E\left(l\left(L(\hat{F}(X))\right)\right)$ is the expected length of the sequence of channel codewords associated with $X$, is an upper bound on the distortion-rate curve $\tilde{D}(R)$, where $P\left(\hat{X} | X\right)$ belongs to the set of all possible transition probabilities such that the average mutual information $I(X; \hat{X}) \leq R$. The convex hull of the operational distortion-rate curve can be found by minimizing the functional

$$J(F^e, F^d, L) = E\left(d(X, F^d(F^e(X)))\right) + \lambda E\left(l(L(F^e(X)))\right),$$

where $\lambda$ can be interpreted as the slope of a line supporting the convex hull of the operational distortion-rate function $\tilde{D}(R)$.

Given arbitrary but fixed analysis/synthesis transformations, there are three necessary conditions for optimality of the $M$ subband quantizers. First, the $m$th band fixed-length encoder mapping $E^e_m$ ($1 \leq m \leq M$) must be such that the product fixed-length encoder mapping $E^e = (E^e_1, \ldots, E^e_m, \ldots, E^e_M)$ satisfies

$$E\left(d(X, F^d(E^e(X^a)))\right) + \lambda E\left(l(L(E^e(X^a)))\right) = \inf_{E} E\left(d(X, F^d(E(X^a)))\right) + \lambda E\left(l(L(E(X^a)))\right).$$

In the above equation, the asterisk "*" denotes optimality. Second, the $m$-band decoder mapping $D^*_m$ must satisfy

$$E\left(d(Z_m, D^*_m(j_m))\right) = \inf_{D_m} E\left(d(Z_m, D_m(j_m))\right),$$

where $Z_m$ is such that $\Phi_m = S(Z_m)$, and $\Phi_m = X - \sum_{n \neq m} F^d_n(j_n)$. Third, the $m$th band variable-length mapping $L^*_m$ must satisfy

$$E\left(l\left(L^*_m\left(\Phi_m\left(F^e(X)\right)\right)\right)\right) = \lim_{n \rightarrow \infty} H(j_n^0, j_{n-1}^0, \ldots, j_1^0, j_{n-1}^1, \ldots, j_M^1, j_1^2, \ldots, j_1^{n-2}, \ldots, j_1^1, \ldots, j_M^1).$$

A detailed derivation of the above equations can be found in [14].

### 3 The Design Algorithm

The design algorithm proposed in this framework is an iterative design algorithm that attempts to simultaneously satisfy the conditions given by equations (1), (2), and (3). Given a fixed Lagrangian value or control parameter $\lambda$, the design algorithm attempts to optimize collectively all subband quantizers to minimize the reconstruction error over all training data subject to a constraint on the entropy rate of the subband/quantization system. This algorithm is shown in [14] to converge to, at least, a locally optimal solution. While the complexity of this design algorithm may be exorbitant, good complexity/performance tradeoffs can be obtained by employing constrained or structured quantizers and carefully choosing the design parameters. This algorithm can be used in conjunction with high-order entropy coders to achieve very good reconstruction at relatively low bit rates. Although further experimental work remains to be done, preliminary results show that the new subband/quantization design algorithm is a very competitive technique for data compression.

### 4 Results and Conclusions

To illustrate the potential of this approach, we employ the proposed design algorithm to design a subband coder which we use to code a variety of images. In this system, the analysis/synthesis filters used are the computationally efficient IIR allpass polyphase filter banks based on two-band decompositions [15]. The subband decomposition considered here is a rectangularly separable one in which the rows are first split into highpass and lowpass sequences and then the columns of the result are split in the same way. Four subband images result from this process, each with a distinct region of the frequency domain associated with it. Each
frequency band is further subdivided into four sub-bands and so on, until a specified number of uniform subbands is reached. Through a large number of experiments, we concluded that a 3-level decomposition (or 64 subbands) is a good compromise between speed of the analysis/synthesis transformations and quantizer complexity.

Figure 3 shows an original test image “Kevin.” Figure 5 shows the same image coded at a relatively low rate of 0.19 bits/pixel. The PSNR is 39.04 dB. Figures 4 and 6 illustrate a high rate example. Figure 4 is the original image “Stephen.” Figure 6 shows Stephen coded at 0.45 bits/pixel. The subjective quality is perceptually identical to the original. The PSNR is 42.40 dB.

The use of these optimality conditions in the construction of the subband coder results in better performance than the isolated design of the subband quantizers. Work continues towards improving the computational efficiency and attaining the best compromise between computational complexity, memory requirements, and performance.

References


Figure 3: The original image “Kevin” at 8 bpp

Figure 4: The original image “Stephen” at 8 bpp

Figure 5: The image “Kevin” coded at 0.19 bpp using the new algorithm. PSNR is 39.04 dB.

Figure 6: The image “Stephen” coded at 0.45 bpp using the new algorithm. PSNR is 42.40 dB.
LAPPED RVQ AND ALPHABET AND ENTROPY CONSTRAINTS

Francesco Nesci, Faouzi Kossentini and Mark J. T. Smith

School of Electrical Engineering, Georgia Institute of Technology, Atlanta Ga 30332

ABSTRACT
This paper reports on some methods to improve the performance of residual vector quantization (RVQ) for low bit rate image coding. The problem addressed is the direct application of RVQ-based techniques to coding images at low bit rates. In the first part of the paper, RVQs are investigated where vectors are allowed to overlap. The RVQ structure allows this to be done in stages and without increasing the bit rate, resulting in a reduction in blocking effects. In the second part of the paper, entropy-constrained RVQ is considered where an additional constraint is placed on the RVQ codebook to improve the reproduction quality given fixed memory constraints.

1. INTRODUCTION
Vector Quantization or VQ is a powerful method for image compression, particularly when used in conjunction with other coding schemes such as subband coding, DPCM, and transform coding. Variations of VQ have also been shown recently to be competitive in isolation. Recent advances [1, 2, 3, 4] in the area of multistage residual vector quantization (RVQ), in particular entropy-constrained RVQ (EC-RVQ), have contributed to significant improvements in coding quality while still maintaining relatively low complexity and memory requirements. [3]. The competitive performance achieved by RVQ comes from the incorporation of the entropy constraint into the RVQ design process. Two new concepts are explored in this paper, both of which are aimed at improving the performance at low bit rates.

In the next section, the notion of overlapping the vector blocks by some amount so that the edge effects are averaged out is investigated. The multistage structure of RVQ is exploited so that distortions, when they appear, are potentially more pleasing to the eye. Following this discussion, the concept of alphabet and entropy constrained RVQ (AEC-RVQ) is investigated. The idea of an alphabet and entropy constraint, which was introduced in [5], is developed and evaluated in the context of RVQ. This approach uses entropy coding with an additional constraint on the RVQ codebook to achieve greater efficiency in terms of memory requirements. The alphabet constraint allows for higher peak bit rates to be used in the design of the coder, thereby improving the performance for the same memory constraints. The paper concludes with a discussion of experimental results.

2. LAPPED RVQ
At low bit rates VQ introduces distortions that tend to appear as blocking artifacts, which noticeably degrade the subjective quality of the image. Some earlier work with overlapping vectors (but in the context of conventional VQ) was reported previously [6] and was shown to result in improvements. This type of overlapping resulted in an appreciable increase in the bit rate which reduced the effective gains achieved. The RVQ structure, however, does not have this problem of bit rate increase due to overlapping. As will be shown, quality improvement can be achieved while maintaining the bit rate or only increasing it slightly, depending on the method of vector selection. This technique used here is called lapped RVQ (LRVQ).

The conventional RVQ approach has been to segment the input into contiguous blocks and to quantize each block with the RVQ. LRVQ is different in the sense that the vectors in each stage of the RVQ have a spatial shift associated with them. In the first stage of the LRVQ, the image is quantized on a block by block basis as usual. Then the complete residual image is reconstructed. That residual image is used as the input to the next stage and new vectors are extracted such that the vector block boundaries are shifted with respect to the boundaries of the previous stage. This process is repeated for all stages of the RVQ. The consequence of changing the vector block segmentation is that the edges between boundaries from stage to stage are no longer coincident and that the blocking effects are smoothed out.

2.1. LRVQ ALGORITHM
Conceptually LRVQ is easy to understand and implement, but some performance issues must be addressed.
which, on the surface, are not obvious. The characteristic of the operational distortion-rate profile for RVQ shows that each subsequent stage contributes less to the distortion than its predecessor. For LRVQ to work well, it is necessary to even out the distortion contributions of the stages. This issue may be addressed in the RVQ design phase by allowing the number of vectors per stage to vary in the design process.

The next issue to address is the pattern for segmenting the residual images into vector blocks, and there are many possibilities. First, one can assume that the image is replicated at all borders such that the left and right edges and top and bottom edges touch. With this partitioning, the number of vectors does not change as a function of the segmentation pattern. Thus the bit rate is the same for LRVQ as it is for RVQ, and all of the LRVQ vectors are complete. The drawback is that when vector blocks straddle the image boundaries, they tend to have large creases or seams due to boundary discontinuous. An approach that avoids this problem is to symmetrically extend the image so that across each of the four image borders lies the corresponding mirror image. This eliminates the seams in the border vectors, but it causes the number of vectors in the residual images to change as a function of the stage in the quantizer. If an LRVQ is being implemented that shifts the blocks by half a vector along each axis at each stage, and if the image is segmented into \( N \times N \) vectors at the first stage, then at the second stage there will be \( (N + 1) \times (N + 1) \) vectors. This will also cause a slight increase in the bit rate, but for images that are 512 x 512 pixels the increase is only about 1% - 2%. Alternatively, the image may be broken up into two different kinds of vectors, interior or full vectors and border or fractional vectors. This method has the same effect on the bit rate as the previous method, but has the benefit of not trying to match contrived or unnatural vectors. It requires the creation of vectors of varying dimension, and therefore changes the way vectors are matched to codebook entries. Each vector extracted from a residual image will have a mask vector of 1's and 0's that indicates which elements in the vector come from the residual and which elements are undetermined. All full vectors will have masks that are all 1's, and only fractional vectors will have masks with 0's. The masks and the vectors are then matched to the codebook entries using a weighted distortion function of the form:

\[
d(x_p, z_p) = \sum_{i=1}^{k} m_p^i (x_p^i - z_p^i)
\]

where \( x_p \) is the \( p \)th-stage residual vector, \( m_p^1, m_p^2, \ldots \) forms its mask, \( z_p \) is the \( p \)th-stage coded vector, and \( k \) is the vector size.

The design algorithm proceeds as follows. The number of stages and a lower bound on the final distortion (here expressed in peak signal-to-noise-ratio or PSNR), when encoding the training images, are specified at the start of the design. Then for each stage, a target PSNR and an initial number of vectors are determined. The LRVQ is then trained in the conventional manner. After the training is completed, the PSNR is calculated and compared to the target PSNR. If it is less than the target, the number of code vectors for the stage is doubled and the training is repeated. This is continued until the PSNR is greater than or equal to the target PSNR. Once the current stage has been designed, the next stage is processed in a similar manner.

When starting a new stage, the algorithm sets the initial number of vectors equal to the final number of vectors from the previous stage and the target PSNR is given by,

\[
PSNR_k = PSNR_{k-1} + \frac{PSNR_M - PSNR_{k-1}}{M - k + 1},
\]

where \( PSNR_k \) is the target PSNR for the current stage, \( M \) is the desired number of stages in the quantizer, and \( PSNR_M \) is the lower bound PSNR for the quantizer. The first stage is a special case and for it the number of vectors is initially set to 2 and the PSNR of the previous stage is calculated by using the training set energy. The best performance results were achieved using a large number of stages which made it easier to distribute the distortion more evenly across the stages. The RVQ design algorithm described in [1, 2] is directly extensible to the case of lapped stage vectors.

3. ALPHABET- AND ENTROPY-CONSTRAINED RVQ (AEC-RVQ)

Critical to the competitiveness of a VQ compression scheme is the effective use of entropy coding. Next, we consider an efficient method for incorporating the entropy constraint in the RVQ design. This is the second phase of the work reported here, which is also aimed at improving the reproduction quality at low rates. This method, which we call alphabet and entropy constrained RVQ (AEC-RVQ) operates by optimally choosing sub-codebooks from a large generic RVQ codebook. The AEC-RVQ algorithm is similar to AEC-VQ [5] in the sense that both operate with one codebook for all rates. However, the new AEC-RVQ typically requires only a fraction of the memory and its encoding is many times faster than that of AEC-VQ. Moreover, its performance is better and it is suitable for real-time implementation.

Embedding the entropy constraint into the RVQ codebook design algorithm is achieved by minimizing
the Lagrangian

\[ J_\lambda = E[d(x_1, \hat{x}_1)] + \lambda E[l(x_1)], \]  

(1)

composed of a distortion term \( d(x_1, \hat{x}_1) \) and a variable codeword length term, \( l(x_1) \). The \( \lambda \) has an interpretation as the slope of the line supporting the convex hull of the operational rate-distortion curve. Each iteration of the algorithm tries to satisfy simultaneously the necessary conditions for optimality derived in [4]. Starting with \( \lambda = 0 \) (which corresponds to the fixed rate RVQ design algorithm [1]), the EC-RVQ algorithm uses a pre-determined sequence of \( \lambda \)'s to design locally optimal variable rate RVQ codebooks. Performance is enhanced by making the rate of the fixed rate RVQ, or the peak bit rate, very high relative to the operating bit rates. A more detailed explanation of the algorithm can be found in [3, 4]. The new constraint considered here is to restrict each codebook to be a subset of a single RVQ codebook for all rates. Thus the design procedure that minimizes equation (1) can throw away some of the stage code vectors (leading to codebooks with smaller sizes) but cannot alter any of the stage code vectors. While the decoder optimization step is a little more complicated, the encoding complexity associated with the AEC-RVQ encoder optimization step can be made very efficient by using \( M \)-search and saving the best \( M \) paths (for each input vector) of the previous iteration for the next iteration. In practice, small values of \( M \) work very well. Hence the complexity required to achieve good performance is low.

4. EXPERIMENTAL RESULTS

Simulation results show that improvement can be achieved by utilizing both the lapping and AEC ideas. In the first set of experiments, an LRVQ was designed with a target PSNR of 29 dB. A total of 250,000 vectors were extracted from a database of head and shoulders images. The test image Lena, shown in Figure 1 is never included in the training set. The resulting LRVQ had 6 stages with \((2, 4, 8, 8, 8, 8)\) vectors at each stage. The vectors were of size \(8 \times 8\) and lapping between stages was based on an offset by 4 pixels in both the horizontal and vertical directions. The test image Lena was then encoded at a bit rate of 0.238 bpp. This image is shown in Figure 2a. As a comparison, the same training set was used to create a conventional jointly optimized RVQ with the same number of stages, but with \((4, 4, 4, 8, 8, 8)\) vectors per stage. Lena was encoded using this conventional RVQ at a bit rate of 0.234 bpp. It is shown in Figure 2b. As expected, both images are of poor quality, because the conventional RVQ is a fixed rate system and the bit rate is relatively low. The conventional RVQ shows the traditional blocking artifacts, while the LRVQ has a different noise characteristic. LRVQ blocking effects seen from the LRVQ are smaller and less noticeable. Reduction in the blockiness can be controlled by choice of the block size, lapping offset, and by more evenly distributing the distortion across all stages in the RVQ.

In a second set of experiments, images were coded at low bit rates with AEC-RVQ. Given the fundamental constraint on memory, AEC-RVQ is able to utilize higher bit rates than EC-VQ, resulting in superior performance. To illustrate this, examine Figure 3, which shows Lena coded at 0.145 bits/pixel using \(8 \times 8\) vectors. Such quality is unusually high for VQ based schemes at this bit rate.

5. CONCLUSIONS

The concepts of lapping and alphabet and entropy constrained RVQ are two approaches for improving the reproduction quality at very low bit rates. Lapping achieves this by explicitly addressing the source of the blocking distortion and attempting to reduce its severity by a stage averaging effect. LRVQ is clearly in its early stages of development and the improvement in performance is only marginal. However, many opportunities remain for improving its performance. One issue is the efficient incorporation of \( M \)-search within the lapped structure. Another area for improvement is in the design of stages with more even contribution to the distortion. The AEC-RVQ approach, also introduced in this paper, attempts to improve quality by utilizing entropy coding and a high peak bit rate. AEC-RVQ was shown to result in significant improvement over conventional VQ methods such as EC-VQ.
Figure 2: The Lena image coded using (a) LRVQ and (b) conventional jointly optimized RVQ. The bit rate and PSNR are (a) 0.238 bpp and 28.2 dB and (b) 0.234 bpp and 27.8 dB.

Figure 3: The Lena image coded with AEC-RVQ at 0.145 bpp. The PSNR is 30.09 dB.

because it allows for much higher peak bit rates.

6. REFERENCES


# NATIONAL SCIENCE FOUNDATION
## FINAL PROJECT REPORT

### PART I - PROJECT IDENTIFICATION INFORMATION

<table>
<thead>
<tr>
<th>1. Program Official/Org.</th>
<th>John Cozzens - NIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Program Name</td>
<td>CIRCUITS AND SIGNAL PROCESSING PROGRAM</td>
</tr>
<tr>
<td>3. Award Dates (MM/YY)</td>
<td>From: 04/52 To: 09/55</td>
</tr>
<tr>
<td>4. Institution and Address</td>
<td>GA Tech Res Corp - E11 Administration Building Atlanta GA 30332</td>
</tr>
<tr>
<td>5. Award Number</td>
<td>9116113</td>
</tr>
<tr>
<td>6. Project Title</td>
<td>Dynamic Multirate Systems for Image and Video Compression</td>
</tr>
</tbody>
</table>

This Packet Contains
NSF Form 98A
And 1 Return Envelope
NSF Grant Conditions (Article 17, GC-1, and Article 9, FDP-11) require submission of a Final Project Report (NSF Form 98A) to the NSF program officer no later than 90 days after the expiration of the award. Final Project Reports for expired awards must be received before new awards can be made (NSF Grants Policy Manual Section 677).

Below, or on a separate page attached to this form, provide a summary of the completed projects and technical information. Be sure to include your name and award number on each separate page. See below for more instructions.

PART II - SUMMARY OF COMPLETED PROJECT (for public use)

The summary (about 200 words) must be self-contained and intelligible to a scientifically literate reader. Without restating the project title, it should begin with a topic sentence stating the project's major thesis. The summary should include, if pertinent to the project being described, the following items:

- The primary objectives and scope of the project
- The techniques or approaches used only to the degree necessary for comprehension
- The findings and implications stated as concisely and informatively as possible

PART III - TECHNICAL INFORMATION (for program management use)

List references to publications resulting from this award and briefly describe primary data, samples, physical collections, inventions, software, etc. created or gathered in the course of the research and, if appropriate, how they are being made available to the research community. Provide the NSF Invention Disclosure number for any invention.

I certify to the best of my knowledge (1) the statements herein (excluding scientific hypotheses and scientific opinion) are true and complete, and (2) the text and graphics in this report as well as any accompanying publications or other documents, unless otherwise indicated, are the original work of the signatories or of individuals working under their supervision. I understand that willfully making a false statement or concealing a material fact in this report or any other communication submitted to NSF is a criminal offense (U.S. Code, Title 18, Section 1001).

Principal Investigator/Project Director Signature: [Signature]
Date: 4-18-97

IMPORTANT:
MAILING INSTRUCTIONS
Return this entire packet plus all attachments in the envelope attached to the back of this form. Please copy the information from Part I, Block I to the Attention block on the envelope.
PART II - SUMMARY OF COMPLETED PROJECT (for public use)

The research performed under this grant focused on new directions in the areas of multirate representations and vector quantization with particular application to image and video compression. Several innovative approaches were studied and developed. Among these are: spatially adaptive FIR and IIR filter banks, where the filter banks can change in response to the incoming signal; efficient high dimension vector quantizers, which allow nonlinear dependencies to be exploited over a wide region of support; entropy-constrained residual vector quantizers, which outperform the current standard JPEG algorithm in terms of SNR and subjective quality, and rate-distortion based approaches to image and video coding. The best of these algorithms are the subband coding methods developed for image and video compression. They use a rate-distortion measure in conjunction with an optimized design process. Performance results surpass the current standards for both low bit rate and high bit rate operation.
PART III - TECHNICAL INFORMATION (for program management use)


The data requested below are important for the development of a statistical profile on the personnel supported by Federal grants. The information on this part is solicited in response to Public Law 99-383 and 42 USC 1885C. All information provided will be treated as confidential and will be safeguarded in accordance with the provisions of the Privacy Act of 1974. You should submit a single copy of this part with each final project report. However, submission of the requested information is not mandatory and is not a precondition of future award(s). Check the "Decline to Provide Information" box below if you do not wish to provide the information.

Please enter the numbers of individuals supported under this grant. Do not enter information for individuals working less than 40 hours in any calendar year.

<table>
<thead>
<tr>
<th>Category</th>
<th>Senior Staff</th>
<th>Post-Doctors</th>
<th>Graduate Students</th>
<th>Under-Graduates</th>
<th>Other Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Total, U.S. Citizens</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B. Total, Permanent Residents</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U.S. Citizens or Permanent Residents²</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>American Indian or Alaskan Native</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asian</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Black, Not of Hispanic Origin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hispanic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pacific Islander</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>White, Not of Hispanic Origin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C. Total, Other Non-U.S. Citizens</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify Country</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Tunisia</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Hong Kong</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Romania</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D. Total, All participants (A + B + C)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disabled³</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

³ Decline to Provide Information: Check box if you do not wish to provide this information (you are still required to return this page along with Parts I-III).

² Use the category that best describes the ethnic/racial status to all U.S. Citizens and Non-citizens with Permanent Residency. (If more than one category applies, use the one category that most closely reflects the person's recognition in the community.)

³ A person having a physical or mental impairment that substantially limits one or more major life activities; who has a record of such impairment; or who is regarded as having such impairment. (Disabled individuals also should be counted under the appropriate ethnic/racial group unless they are classified as "Other Non-U.S. Citizens.")

AMERICAN INDIAN OR ALASKAN NATIVE: A person having origins in any of the original peoples of North America and who maintains cultural identification through tribal affiliation or community recognition.

ASIAN: A person having origins in any of the original peoples of East Asia, Southeast Asia or the Indian subcontinent. This area includes, for example, China, India, Indonesia, Japan, Korea and Vietnam.

BLACK, NOT OF HISPANIC ORIGIN: A person having origins in any of the black racial groups of Africa.

HISPANIC: A person of Mexican, Puerto Rican, Cuban, Central or South American or other Spanish culture or origin, regardless of race.

PACIFIC ISLANDER: A person having origins in any of the original peoples of Hawaii; the U.S. Pacific territories of Guam, American Samoa, and the Northern Marinas; the U.S. Trust Territory of Palau; the islands of Micronesia and Melanesia; or the Philippines.

WHITE, NOT OF HISPANIC ORIGIN: A person having origins in any of the original peoples of Europe, North Africa, or the Middle East.