# Error Resilient Pyramid Vector Quantization for Image Compression

Andy C. Hung, Ely K. Tsern, and Teresa H. Meng

**Transactions on Image Processing regular paper IP-1310.**

Please address all editorial correspondence for the authors to:

**Andy C. Hung**
**12849 Canario Way**
**Los Altos Hills, CA 94305**

**Tel: (415) 949-5529 Fax: (408) 752-9101**
**E-mail: achung@cs.stanford.edu**

**EDICS category:** 1.1, Image Processing – Coding.

**Keywords:** pyramid vector quantization, lattice vector quantization, subband coding, image compression, channel optimization, error resilience.

BIO Modification for Andy C. Hung

Andy C. Hung (S'90 - M'95) received the B.S. degree from the Massachusetts Institute of Technology in 1987, the M.S. degree from the University of California, Berkeley in 1988, and the Ph.D. degree from Stanford University in 1995.

From 1994 to 1998, he worked at Chromatic Research, where he was manager of the MPEG DVD project. Currently, he is working at InterVideo, Inc. on fast video algorithms for personal computer applications.

# Error Resilient Pyramid Vector Quantization for Image Compression

Andy C. Hung[1], *Member IEEE*, Ely K. Tsern[2], *Member IEEE*, and Teresa H. Meng[3], *Senior Member IEEE*

*Abstract*- **Pyramid vector quantization (PVQ) uses the lattice points of a pyramidal shape in multidimensional space as the quantizer codebook. It is a fixed-rate quantization technique that can be used for the compression of Laplacian-like sources arising from transform and subband image coding, where its performance approaches the optimal entropy-coded scalar quantizer without the necessity of variable length codes. In this paper, we investigate the use of PVQ for compressed image transmission over noisy channels, where the fixed-rate quantization reduces the susceptibility to bit-error corruption. We propose a new method of deriving the indices of the lattice points of the multidimensional pyramid and describe how these techniques can also improve the channel noise immunity of general symmetric lattice quantizers. Our new indexing scheme improves channel robustness by up to 3 dB over previous indexing methods, and can be performed with similar computational cost. The final fixed-rate coding algorithm surpasses the performance of typical Joint Photographic Experts Group (JPEG) implementations and exhibits much greater error resilience.**

## 1 Pyramid Vector Quantization

Pyramid vector quantization (PVQ) was introduced by Fischer [1][2] as a fast and efficient method of quantizing Laplacian-like data, such as generated by transforms or subband filters [3]-[6] in an image compression system. PVQ has very simple systematic encoding and decoding algorithms and does not require significant codebook storage. It combines the robustness of fixed-rate codes with the performance of entropy-coded scalar quantization. Considerable research in PVQ algorithms has culminated in high

changed companies

[1] Andy Hung is with InterVideo Inc., Fremont, CA 94539 (e-mail: achung@cs.stanford.edu).

[2] Ely Tsern is with Rambus Inc., Mountain View, CA 94040 (e-mail tsern@rambus.com).

[3] Teresa Meng is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 (e-mail: meng@mojave.stanford.edu).

[4] This research was sponsored by the Joint Services Electronics Program, US Army Research Office, contract DAAH04-94-G-0058, and by the Advanced Research Projects Agency, contract DABT63-91-K-0002.

add acknowledgment

performance, error resilient PVQ image compression systems for both transforms [7]-[14] and subband decompositions [15]-[24]. PVQ has also been implemented in hardware [18]-[20], taking advantage of a computationally constructed codebook instead of a stored codebook to build a robust, low power, video-rate PVQ decoder.

In this paper, we propose new ways of assigning indices to the points in the PVQ codebook that improves channel robustness by up to 3 dB over previous enumerations, and up to 6 dB over a randomly enumerated codebook. These new indexing techniques require roughly the same encoding and decoding hardware complexity as previous enumerations and can be applied to any of the fixed rate PVQ coding systems described previously in the literature. We discuss the theoretical and simulated advantage of new techniques through channel noise models of the PVQ indices and codebooks. Finally, we show the practical advantages of PVQ by demonstrating an error-resilient PVQ system that exceeds the performance of Joint Photographic Experts Group (JPEG) implementations, both with and without channel error.

add extra word to make “JPEG” make sense

## 2 A Brief Overview of Pyramid Vector Quantization

Pyramid vector quantization takes its name from the geometric shape of the points in its codebook. It is designed for Laplacian random variables, whose equiprobable contours form multidimensional pyramids. This can be seen from the multidimensional Laplacian probability density:

$$f_X(\boldsymbol{x}) = (\lambda/2)^l e^{-\lambda \sum_{i=1}^{l} |x_i|} , \qquad (1)$$

where $\boldsymbol{x}$ is a vector of length $l$. The surfaces of equal probability is defined in an $l$ dimension space where the $l_1$ norm of $\boldsymbol{x}$ is a constant, $r$, representing the radius of the surface

$$\sum_{i=1}^{l} |x_i| = r . \qquad (2)$$

The surface given by a fixed $l_1$ constraint on the coordinates, equation (2), is called a pyramid. In the literature, the $l_1$ surface is often referred to by other names: dipyramid or bipyramid, by the shape in three dimensions; generalized octahedron, from the corresponding Platonic solid; cross polytope because of the vertex locations; and co-cube, since it is the polyhedral dual of the cube structure upon exchange of vertex locations and face normals[25][26][27].

The radial distribution for $r$ can be obtained from the convolution of $l$ exponential distributions; it is an Erlang distribution,

$$f_R(r) = \frac{\lambda^r r^{l-1} e^{-\lambda r}}{(l-1)!}. \tag{3}$$

The PVQ codebook, $P(l, k)$, consists of the set of integer vectors of length $l$ whose absolute values sum to $k$: $P(l, k) = \left\{ \boldsymbol{x} : x_i \in Z \text{ and} \sum |x_i| = k \right\}$, as shown in figure 1 in two and three dimensions. It can be obtained by intersecting the pyramid surface of integer radius $r = k$, described by equation (2), with the cubic lattice, resulting in a regular distribution of codebook points on the pyramid surface. The PVQ codebook, $P(l, k)$, is typically scaled to fit the desired pyramid contour. For very large vector dimensions, $l$, PVQ has asymptotic properties: by the Central Limit Theorem, the radial distribution $f_R(r)$ approaches that of a Gaussian; by the Asymptotic Equipartition Theorem [1], the vector $\boldsymbol{x}$ clusters uniformly on the pyramid of radius $r = l/\lambda$.

missing in text

This section has covered the geometric structure of the PVQ codebook. The following two sections describe the enumeration of the pyramid codebook $P(l, k)$ – the method which assigns a transmittable index to each PVQ codebook vector.

# 3 Magnitude PVQ Enumeration

Fischer introduced the first PVQ enumeration technique [1] that showed the feasibility of assigning unique indices to the pyramid codebook by computation. We shall refer to Fischer’s original technique as magnitude enumeration in this paper, for reasons that will later become clear. Magnitude enumeration serves as an excellent starting point to introduce PVQ enumeration, so we devote some time to its description in this section, showing some simplifications to the existing enumeration equations. In later sections, we will use the same terminology to introduce and develop our own enumeration techniques.

Enumeration assigns a unique index to all possible vectors in the PVQ codebook, $P(l, k)$, imparting a sorting order to the PVQ codebook vectors. For example, magnitude enumeration sorts each vector based on the *magnitude* of each of its elements. The first two vectors in the codebook $P(l, k)$ begin with $l-1$ zeroes: index 0 corresponds to the vector $(0, 0, \ldots, k)$ and index 1 corresponds to the vector $(0, 0, \ldots, -k)$. Systematic sorting for enumeration is done through counting formulas for the number of vectors in the pyramid; this is a common concept to all pyramid enumerative techniques.

The number of vectors in the pyramid codebook $P(l, k)$ is denoted by $N(l, k)$. This is related to the binary codeword index length, which is $\lceil \log_2 N(l, k) \rceil$ bits. $N(l, k)$ can be viewed as the number of ways

ceiling, not floor operator

$l$ integer values in a vector can have an absolute sum of $k$. Of these $N(l, k)$ possible combinations, only $N(l-1, k-|i|)$ of them result in the value $i$ as the first element (for $i$ an integer between $-k$ and $k$), by the definition of $N(l, k)$. Hence, $N(l, k)$ is the sum of the number of ways of obtaining a pyramid codebook vector starting with $i$, for every possible value of $i$. This is written as

$$N(l, k) = \sum_{i=-k}^{k} N(l-1, k-|i|) \ , \tag{4}$$

where the boundary conditions are $N(l, 0) = 1$ for all integer $l$, and $N(0, k) = 0$ for integer $k \neq 0$. A recursive application of equation (4) will obtain a value for any $N(l, k)$ .

Equation (4) has a geometric interpretation: If one of the dimensions is fixed for a pyramid in dimension $l$, then the resulting shape is a pyramid in dimension $l-1$. This is shown graphically in figure 1 by fixing the $x_1$ dimension for the pyramid in three dimensions to obtain a pyramid in two dimensions.

Equation (4) has a lexicographic interpretation: Just like the subranges of Webster’s dictionary are ordered in terms of each letter from ‘a’ to ‘z’, the subranges in magnitude enumeration are ordered in terms

missing “1”

of the *magnitude* of each vector element, $i$, by the following sequence: 0, $\boxed{1,}$ −1, 2, −2, …, $k$, $-k$. Suppose $\boldsymbol{x}$ is a vector of length $l$, $\boldsymbol{x} = (x_1, x_2, \ldots, x_l)$ , in the pyramid codebook $P(l, k)$ . Based on the first vector element, $x_1$, the integer range from 0 to $N(l, k) - 1$ can be subdivided into subranges of size $N(l-1, k-|i|)$ for $i$ between $-k$ to $k$, using equation (4). If the first element is $x_1 = 3$, for example, then the subrange consists of the $\boxed{l-1}$ remaining vector elements with an absolute sum of $k-3$, and has size $N(l-1, k-3)$ .

an “l” not “i”

Using the subrange sizes and their order, it is possible to calculate the offset of each subrange from zero in magnitude enumeration. The magnitude enumeration index offset of the subrange consisting of vectors with first element $x_1 = i$ in the codebook $P(l, k)$ is denoted as $O_M(i, l, k)$ . This is calculated from the two first subranges representing $i = 0$ and $i = 1$, $O_M(0, l, k) = 0$ and $O_M(1, l, k) = N(l-1, k)$ , as

$$O_M(i, l, k) = \begin{cases} N(l-1, k) + 2\sum_{j=1}^{i-1} N(l-1, k-j) & \text{for } i \geq 2 \text{, and} \\ O_M(|i|, l, k) + N(l-1, k-|i|) & \text{for } i < 0 \text{.} \end{cases} \tag{5}$$

The offset for positive $i$, in equation (5), is calculated by adding all subranges with a smaller magnitude. The offset for negative $i$, is just the sum of the offset for positive $i$, $O_M(|i|, l, k)$ , and the subrange for positive $i$, $N(l-1, k-|i|)$ . Although equation (5) follows the original definitions given in [1], it is not the simplest definition. With manipulations, either through massaging equation (4), or through application of the volume summation formula, equation (48), it is easy to see that

$$O_M(i, l, k) = \begin{cases} N(l, k) - N(l, k-i) - N(l-1, k-i) & \text{for } i > 0\text{, and} \\ N(l, k) - N(l, k-|i|) & \text{for } i \le 0\,. \end{cases} \tag{6}$$

The magnitude PVQ index of a vector $\boldsymbol{x}$ is obtained from summing the offsets of nested subranges, and is recursively defined from equation (6). Each recursive step removes another element from the vector $\boldsymbol{x}$. Let $x_1$ be the first element of the vector $\boldsymbol{x}$; let $\boldsymbol{x}_{[1]}$ denote the subvector of the vector $\boldsymbol{x}$ excluding the first element. Then the magnitude enumerated index $I_M(\boldsymbol{x}, l, k)$ is recursively defined as

$$I_M(\boldsymbol{x}, l, k) = O_M(x_1, l, k) + I_M(\boldsymbol{x}_{[1]}, l-1, k-|x_1|) \quad , \tag{7}$$

where the "empty" vector of no elements has an index $I_M((\ )\boxed{, 0,}\, 0) = 0$. For example, suppose the input vector of $(1, 0, -1)$ is to be enumerated on the pyramid codebook $P(3, 2)$. Recursive application of equation (7) yields

my mistake, typo, should be 0

$$\begin{aligned} I_M((1, 0, -1), 3, 2) &= O_M(1, 3, 2) + I_M((0, -1), 2, 1) \\ &= O_M(1, 3, 2) + O_M(0, 2, 1) + I_M((-1), 1, 1) \\ &= O_M(1, 3, 2) + O_M(0, 2, 1) + O_M(-1, 1, 1) \quad . \end{aligned} \tag{8}$$

The decoding proceeds as follows. Given a received magnitude enumerated index, $I_m$, in the pyramid codebook, $P(l, k)$, the first vector element $x_1$ is determined from the unique subrange for which $O_M(x_1, l, k) \le I_m < O_M(x_1, l, k) + N(l-1, k-|x_1|)$. Once the first element is decoded, the offset of the subrange is subtracted from the original index, resulting in a number ranging from $0$ to $N(l-1, k-|x_1|) - 1$. The remaining elements now form a vector of length $l-1$ with absolute sum of $k-|x_1|$. This is recursively decoded by the same procedure as before, except the new index is $I'_m = I_m - O_M(x_1, l, k)$, corresponding to the vector of $l-1$ elements in the pyramid codebook $P(l-1, k-|x_1|)$.

Fischer [1] derives a fast method to calculate $N(l, k)$. From equation (4), the relationship between $N(l, k) - N(l, k-1)$ can be simply expressed as

$$N(l, k) - N(l, k-1) = N(l-1, k) + N(l-1, k-1) \quad . \tag{9}$$

This can be rearranged into

$$N(l, k) = N(l, k-1) + N(l-1, k) + N(l-1, k-1) \quad . \tag{10}$$

Since magnitude enumeration converts between pyramid codebook vectors and indices by formulas involving various combinations of $N(l, k)$, the pyramid codebook need not be stored. This means PVQ codebooks can be very large, easily having between $2^{64}$ to $2^{128}$ entries. For maximum efficiency, the values $N(l, k)$ are usually precomputed, resulting in a table of size $lk$, logarithmic in codebook size. Alternate methods of deriving $N(l, k)$ are described in Appendix A.

# 4 Robust Enumeration of the Multidimensional Pyramid

Error resilience can be achieved through permutation of the PVQ codebook indices to minimize the effect of bit-errors on the indices. In general, close Hamming neighbors in the codebook indices should correspond to close spatial neighbors on the pyramid. Since PVQ codebooks are usually far too large to have such a permutation map stored in memory, we investigate new enumeration methods that automatically yield robust indices in this section. The original approach was *magnitude enumeration*, discussed in section 3. The second approach, which we call *linear enumeration*, is a variant on Fischer’s magnitude enumeration and was originally used in a slightly different form by Swaszek [28]. The third and fourth approaches that we develop here use conditional product codes; they are the *conditional product code enumeration* and *conditional product-product code enumeration* techniques, and both have a 3 dB advantage over the magnitude and linear enumerations under random bit errors.

## 4.1 Linear Enumeration

Linear enumeration relies on constructing a better tree structure than the magnitude enumeration. As described before, magnitude enumeration divides the subranges by the magnitude of the each vector element, in the order $0, 1, -1, 2, -2, \ldots, k, -k$. Adjoining subranges which are close in the indexing sense (and Hamming sense) can be far in a spatial sense. Linear enumeration improves the indexing by reordering the subranges in a linear fashion as $-k, -k+1, \ldots, -1, 0, 1, \ldots, k-1, k$, so adjoining subranges correspond to consecutive values.

The subranges of $N(l, k)$ in linear enumeration are ordered by value of the first vector element $x_1 = i$, rather than the magnitude of the first element, as in magnitude enumeration. The offset of subrange $i$ of the first element of vector $\boldsymbol{x}$ for linear enumeration is denoted as $O_L(i, l, k)$ and is defined for $-k < i \le k$ by the following equation:

$$O_L(i, l, k) = \sum_{j=-k}^{i-1} N(l-1, k-|j|) \ , \tag{11}$$

and by $O_L(i, l, k) = 0$ for $i = -k$. Through application of the volume summation formula, equation (48), the above equation can be simplified to

$$O_L(i, l, k) = \begin{cases} (N(l, k - |i-1|) + N(l-1, k - |i-1|))/2 & \text{for } i \le 1 \text{, and} \\ N(l, k) - (N(l, k-i) + N(l-1, k-i))/2 & \text{for } i \ge 1 \text{.} \end{cases} \tag{12}$$

In equation (12), enumeration for $i = -k$ requires a negative $k$ index in $N(l, k)$, which should be treated as zero.

The PVQ index of a vector $\boldsymbol{x}$ can be recursively defined similarly to that of magnitude and linear enumeration. Each recursive step removes the first element from the vector $\boldsymbol{x}$. Let $x_1$ be the first element of the vector $\boldsymbol{x}$; let $\boldsymbol{x}_{[1]}$ denote the subvector of the vector $\boldsymbol{x}$ excluding the first element. Then the index $I_L(\boldsymbol{x}, l, k)$ is recursively defined as

$$I_L(\boldsymbol{x}, l, k) = O_L(x_1, l, k) + I_L(\boldsymbol{x}_{[1]}, l-1, k - |x_1|) \quad . \tag{13}$$

## 4.2 Conditional Product Code Enumeration

Unlike linear and magnitude enumerations, which divide the subranges by the value of the vector elements, conditional product code enumeration forms subranges based on the number of non-zero (or *significant*) elements in the vector. Since each significant element is symmetric in distribution, each has a corresponding sign bit. These sign bits can be packed into the least significant bits of the index. As long as the number of significant elements can be correctly decoded from the corrupted index, the sign bits form a product code in the least significant bits: They will be independent of each other, limiting the effects of errors. Figure 2 shows the classification of pyramid points based on the number of significant elements.

The indices derived through this enumeration have the *conditional product code* structure. We define a conditional product code in terms of codewords $c_1$, $c_2$, and $c_3$ when the codewords $c_2$ and $c_3$, whose size and length may be conditional on the reception of $c_1$, form a binary product code *and* the code $c_1c_2c_3$ has fixed length $l_1 + l_2 + l_3$, though neither $l_1$, $l_2$, nor $l_3$ need be fixed length individually. The conditional product code property is extremely useful for high-rate pyramids where $1 < l \ll k$ because the sensitive sign bit information has been extracted and isolated.

Conditional product enumeration requires four quantities, each dependent on the number of significant elements, $s$. The first quantity is the significant elements term, $R(s, l, k)$, which is the number of pyramid points out of $N(l, k)$ with $s$ significant elements. The second quantity is the pattern distribution term, $D(s, l)$, which is the number of patterns that distribute $s$ significant elements over a vector length of $l$. The third quantity is the shape term, $S(s, k)$, which is the number of ways to add $s$ positive integers to the value of $k$. The fourth quantity is the sign bits term, $B(s)$, which is the number of ways to assign signs to $s$ non-zero integers.

$N(l, k)$ can be expressed by the sum of $R(s, l, k)$ from 1 to the maximum number of significant elements, $m = \min(l, k)$. This is

missing from text

$$N(l, k) = \sum_{s=1}^{m} R(s, l, k) \quad . \tag{14}$$

The term $R(s, l, k)$ is the product of the pattern distribution, $D(s, l)$, the shape, $S(s, k)$, and the sign bits term, $B(s)$, giving

$$R(s, l, k) = D(s, l)S(s, k)B(s) \quad . \tag{15}$$

The pattern distribution term $D(s, l)$ is

clarification.

$$D(s, l) = \binom{l}{s} , \tag{16}$$

which can be enumerated by the *binomial addition* identity. This recursive formula states that $D(s, l)$ is the sum of the possibilities that first element is zero (or not significant) and the possibilities that the first element is significant:

$$D(s, l) = D(s, l-1) + D(s-1, l-1) \quad . \tag{17}$$

A unique distribution index $I_D(\boldsymbol{x}, s, l)$ for the pattern of a vector $\boldsymbol{x}$, of length $l$ and with $s$ significant elements, can be derived from the above recursion. This recursion is based on the first element, $x_1$, of the vector $\boldsymbol{x}$ and the remaining elements, the vector $\boldsymbol{x}_{[1]}$, as follows:

$$I_D(\boldsymbol{x}, s, l) = \begin{cases} D(s, l-1) + I_D(\boldsymbol{x}_{[1]}, s-1, l-1) & \text{if } x_1 \neq 0 \\ I_D(\boldsymbol{x}_{[1]}, s, l-1) & \text{if } x_1 = 0, \end{cases} \tag{18}$$

with $I_D(\boldsymbol{x}, 0, l) = 0$.

remove “for any x”, my mistake.

The shape term $S(s, k)$ describes the number of shapes of a vector as

$$S(s, k) = \binom{k-1}{s-1} . \tag{19}$$

It is the number of ways $s$ positive integers sum to $k$, and can be enumerated on the first positive integer by the *upper index summation* binomial identity, where the enumeration is based on the value of the first positive integer and the resulting shapes of one fewer dimension, $s - 1$:

$$S(s, k) = \sum_{i=1}^{k-(s-1)} S(s-1, k-i) \quad . \tag{20}$$

There are only $k - (s - 1)$ possible values for the first positive integer, since each of the $s$ integers must be *greater* than zero as well as sum to $k$. For enumeration based on the first significant element, the corresponding offset for the significant element equaling $i$ is determined for $1 \le i \le k - s + 1$ as

$$O_S(i, s, k) = \sum_{j=1}^{|i|-1} S(s-1, k-j) = S(s, k) - S(s, k - |i| + 1) \quad , \tag{21}$$

where the last equality is derived through application of the upper summation binomial identity. The corresponding index into the shape field $I_S(\boldsymbol{x}, s, k)$ is determined as

$$I_S(\boldsymbol{x}, s, k) = \begin{cases} O_S(|x_1|, s, k) + I_S(\boldsymbol{x}_{[1]}, s-1, k - |x_1|) & \text{if } x_1 \neq 0 \\ I_S(\boldsymbol{x}_{[1]}, s, k) & \text{if } x_1 = 0, \end{cases} \tag{22}$$

with $I_S(\boldsymbol{x}, s, s) = 0$. replace with this, my mistake.

The term $B(s)$ is the number of ways to assign signs to $s$ non-zero elements. There is exactly one sign bit per significant element; hence,

$$B(s) = 2^s . \tag{23}$$

The simplest way of constructing $B(s)$ is to concatenate the sign bits of each significant element of the vector going from the first significant element in the vector to the last. This results in a string exactly $s$ bits in size. We can express this by the following equation for the index $I_B(\boldsymbol{x}, s)$.

$$I_B(\boldsymbol{x}, s) = \begin{cases} 2^{s-1} + I_B(\boldsymbol{x}_{[1]}, s-1) & \text{if } x_1 < 0 \\ I_B(\boldsymbol{x}_{[1]}, s) & \text{if } x_1 = 0 \\ I_B(\boldsymbol{x}_{[1]}, s-1) & \text{if } x_1 > 0, \end{cases} \tag{24}$$

with $I_B(\boldsymbol{x}, 0) = 0$.

Using the above formulas for $D(s, l)$, $S(s, k)$, and $B(s)$, we can reduce equation (14) to the following explicit formula, valid for all positive vector lengths $l > 0$:

$$N(l, k) = \sum_{s=1}^{m} R(s, l, k) = \sum_{s=1}^{m} D(s, l) S(s, k) B(s) = \sum_{s=1}^{m} 2^s \binom{l}{s} \binom{k-1}{s-1} \quad . \tag{25}$$

Since the number of points is unchanged, the codeword index is still $\lceil \log_2 N(l, k) \rceil$ bits long. For robustness, the significant elements subranges, $R(s, l, k)$, are arranged in *reverse order*, from $s = m$ down to $s = 1$. The offset significant elements subrange of value $s$ is given by

$$O_R(s, l, k) = \sum_{i = s+1}^{m} R(i, l, k) \ . \tag{26}$$

Once the subrange representing the number of significant elements $s$ is determined, the pattern, shape, and sign bits can be enumerated, forming a number ranging from $0$ to $R(s, l, k) - 1$. The sign bits are stored as a binary conditional product code in the least significant bits of the codeword, the index of which is $I_B(\boldsymbol{x}, s)$. The shape $S(s, k)$ is enumerated by the upper index summation formula into $I_S(\boldsymbol{x}, s, k)$. The pattern distribution $D(s, l)$ is enumerated by the binomial addition formula into $I_D(\boldsymbol{x}, s, l)$. The product enumeration index $I_P(\boldsymbol{x}, l, k)$ is the combination of the three other indices

$$\begin{aligned} I_P(\boldsymbol{x}, l, k) = {} & O_R(s, l, k) + \\ & (I_D(\boldsymbol{x}, s, l)S(s, k) + I_S(\boldsymbol{x}, s, k))B(s) + I_B(\boldsymbol{x}, s) \ . \end{aligned} \tag{27}$$

To decode the formula for a received product index $I_p$, we identify the unique $s$ for which $O_R(s, l, k) \le I_p < O_R(s - 1, l, k)$. After subtracting $O_R(s, l, k)$ from $I_p$, we obtain $I_p'$ a value between 0 and $R(s, l, k) - 1$. The sign bits index $I_b$ are stored in the least significant $s$ bits, so they may be obtained by right-shifting the index $I_p'$ by $s$ bits. The index now contains values between 0 and $D(s, l)S(s, k) - 1$. The pattern index $I_d$ is the quotient of dividing the index by $S(s, k)$; the shape index $I_s$ is the remainder of the division.

Both the shape and pattern can be recursively decoded by a similar procedure to magnitude and linear enumeration. If division is a problem, the patterns $D(s, l)$ can be stored premultiplied by $S(s, k)$. This increases the enumeration table size by a factor of $k$, though that is insignificant compared to the true size of the PVQ codebook. The information structure of the conditional product code is shown in figure 3.

## 4.3 Conditional Product-Product Enumeration

Conditional product-product enumeration is even more robust than conditional product enumeration described in the previous subsection. It is obtained by forcing the shape $S(s, k)$ term in the enumeration to be a conditional product code with respect to the significant elements bit-field. This is done by allocating a total $2^{\lceil \log S(s, k) \rceil}$ values – a power of 2 – for the $S(s, k)$ possible shape assignments. Now the pattern, shape and sign bits are simply concatenated together, so *both* the shape and sign bits form a conditional binary product code with respect to the significant elements bit-field. This has a slight advantage in error propagation *and* hardware decodability, since no division by $S(s, k)$ is required to obtain the pattern field $D(s, l)$, just a right shift by $\lceil \log_2 S(s, k) \rceil$ bits. The equation for the maximum index of a conditional product-product enumeration of pyramid vectors $P(l, k)$ can be expressed as

$$N_{PP}(l, k) = \sum_{s=1}^{m} D(s, l) 2^{\lceil \log S(s, k) \rceil} B(s) \quad . \tag{28}$$

The new maximum value allowed, $N_{PP}(l, k)$, is greater than $N(l, k)$, adding, on average, half a bit to the enumeration index. The indexing equation is identical to the conditional product code enumeration for the pattern index, $I_D(\boldsymbol{x}, s, l)$, the shape index, $I_S(\boldsymbol{x}, s, k)$, and the sign bits index, $I_B(\boldsymbol{x}, s)$, in equations (18), (22), and (24). However a change is made to the significant elements offset, resulting in

$$O_{R_{PP}}(s, l, k) = \sum_{i = s+1}^{m} D(s, l) 2^{\lceil \log S(s, k) \rceil} B(s) \quad . \tag{29}$$

Then the conditional product-product enumerated index for a vector $x$, $I_{PP}(x, l, k)$, is given by

$$\begin{aligned} I_{PP}(\boldsymbol{x}, l, k) = {} & O_{R_{PP}}(s, l, k) + \\ & (I_D(\boldsymbol{x}, s, l) 2^{\lceil \log S(s, k) \rceil} + I_S(\boldsymbol{x}, s, k)) B(s) + I_B(\boldsymbol{x}, s) \quad . \end{aligned} \tag{30}$$

A breakdown of the index field for a given conditional product-product code is shown in figure 4.

# 5 Channel Error Analysis

All four enumeration techniques described earlier, magnitude, linear, conditional product, and conditional product-product enumeration have properties of tree structured codes, which improve their error resilience over a randomly arranged pyramid codebook. Furthermore, the conditional product and conditional product-product enumerations have product code properties. This section examines these error resilience properties in detail.

## 5.1 Random Enumeration Error

The expected squared error $E_R$ from an index corruption of a randomly arranged codebook is the average distance between any two vectors.

$$E_R = \frac{1}{N(l, k)(N(l, k) - 1)} \sum_{i \neq j} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \quad , \tag{31}$$

where $\boldsymbol{x}_i$ are vectors from the randomly arranged PVQ codebook, $P(l, k)$, and the indices $i$ and $j$ range from $0$ to $N(l, k) - 1$. Since $\boldsymbol{x}_i$ has zero mean, equation (31) can be rewritten as

$$E_R = \frac{2N(l, k)}{N(l, k) - 1} E[\|\boldsymbol{x}\|^2 | l, k] \quad , \tag{32}$$

where $E[\|\boldsymbol{x}\|^2|l, k]$ is the codebook variance, given by equation (52). For high rates, equation (32) can be approximated by $E_R \approx 2E[\|\boldsymbol{x}\|^2|l, k] \approx 4k^2/(l+1)$ , which is twice the codebook variance. The randomly enumerated codebook results in a higher channel error than all four systematic PVQ enumeration techniques, as we shall shortly see.

## 5.2 Magnitude and Linear Enumeration Error

Magnitude and linear enumeration are tree structured techniques, with sequential encoding and decoding of each vector element. A bit error in the PVQ index may corrupt all the decoded vector elements, but generally only affects those elements with an enumerative subrange smaller than the value of the flipped bit; i.e. it only affects the subsequent elements coded past that bit error, much like Huffman (prefix) or Arithmetic (non-prefix) coding. Hence, the approximate probability of corruption increases linearly from the first index bit, with the fraction of elements corrupted with a single bit $j$ error in a $\log N$ bit index as

$$f(j, N) \approx (\log N - j)/\log N \quad . \tag{33}$$

Hence, a randomly selected single bit error, on average, corrupts just *half* of the elements, rather than *all* the elements. This results in a significant robustness advantage of magnitude and linear enumeration over random enumeration. For brevity, we leave more detailed error analysis to [41]; in this subsection, we shall present a simplified model of PVQ vector error due to a single bit index error.

We assume that the information about the significant elements is distributed evenly through the PVQ index. The linear and magnitude index model is shown in figure 5 for a vector with six significant elements, with the major difference between the two in the last two bits of the index, which are usually sign bits in the magnitude enumeration technique.The error energy induced by a single bit corruption at position $j$ in the index is approximately $f(j, N(l, k))E_R$ , where $E_R$ is given by equation (32). As described above, for a large enough vector length and index, a bit error corrupts on average half the elements. This means that the linear and magnitude enumeration error energy is approximately half that of the random enumeration

$$E_L \approx E_M \approx (1/2)E_R \approx 2k^2/(l+1) \quad . \tag{34}$$

This equation corresponds to the high rate derivations from [41], holding for both magnitude and linear enumeration. We should note that the difference between linear enumeration and magnitude enumeration only holds for small vector dimensions, primarily due to the different treatment of the least significant bits and minor differences in equation (33).

## 5.3 Conditional Product and Product-Product Enumeration Error

Conditional product enumeration is designed to place the most important information (in terms of bit-error corrupting effects) into the first few bits. As described in [40], it is most advantageous to code the most important information first in a tree structured enumerated code because correct reception of coded information depends on previously coded information being correct.

For conditional product code enumeration, the significant elements are enumerated first, followed by the pattern, followed by the shape, as shown in figure 3. This results in four classes of error: 1) the error in the number of significant elements $s$ ($R$ field); 2) the error in the pattern ($D$ field); 3) the error in the shape ($S$ field); 4) the error in the sign bits ($B$ field). Each of the four index fields have different error characteristics. We will go through each field, describing the characteristics of single bit error.

A bit error in the $R$ significant elements field is considered catastrophic, regardless of bit-error position within the field. The catastrophic error is approximated by the random enumerated codebook error, s

remove “s” my mistake.

$$2E[\|\mathbf{x}\|^2 \big| l, k] \quad . \tag{35}$$

The error for the $j$th bit within the $D$ distribution field is given by the fraction $f(j, D(s, l))$ of elements incorrect multiplied by the catastrophic error plus the fraction of elements that turn out correct $(1 - f(j, D(s, l)))$ multiplied by the shape error, since the shape field follows the distribution field and must therefore also be corrupted if the distribution field is corrupted. For a bit error at position $j$ within the distribution field

$$f(j, D(s, l))2E[\|\mathbf{x}\|^2 \big| s, l, k] + (1 - f(j, D(s, l)))E[\|\mathbf{x} - \mathbf{x}_S\|^2 \big| s, l, k] \quad , \tag{36}$$

where $E[\|\mathbf{x} - \mathbf{x}_S\|^2 \big| s, l, k]$ is the expected shape error given the vector $\mathbf{x}$ (see equation (58)).

bad equation reference, see appendix

A bit error in the $S$ shape field is also modeled as a tree code. For a bit error at position $j$ within the shape field, this is

bracket instead of paren.

$$f(j, S(s, k))E[\|\mathbf{x} - \mathbf{x}_S\|^2 \big| s, l, k] \quad , \tag{37}$$

where the $f(j, S(s, k))$ is the fraction of corrupted elements.

Finally, any single bit error in the sign $B$ field is given by the sign bit error as

$$E[\|\mathbf{x} - \mathbf{x}_B\|^2 \big| s, l, k] \quad , \tag{38}$$

where this term is given by equation (59). The distortion induced by a sign bit error is independent of the bit position within the field.

bad equation reference, see appendix

Together, equations (35), (36), (37), and (38) form a complete description of the conditional product enumeration error by bit-error position in the index, given $s$ significant elements. As the number of bits in the index grows large, most of the index describes the shape. Thus the high rate error is due to the shape field error, and since the shape field is a tree-structured code, half the error due to total shape corruption:

$$E_P \approx (1/2)E[\|\mathbf{x} - \mathbf{x}_S\|^2 | l, k] \approx k^2/(l+1) \quad , \tag{39}$$

using appropriate high rate approximations and equation (56).

From comparing equations (34) and (39), the channel optimization gain of product enumeration over magnitude or linear enumeration is 3 dB; the gain over random enumeration is 6 dB. This result holds for single bit errors, though clearly it also holds for multiple bit errors so long as the bit errors occur primarily in the shape field of the product-enumerated index.

## 5.4 Geometric Quantization of Shapes and Volumes

Conditional product enumeration applies to any symmetrically defined codebook that is an even function of each vector coordinate. Furthermore, it also applies to volumes defined by a sequence of surfaces. If each surface is indexed by $i$, then all individual $R_i(s, l, k)$ should be sorted in terms of the significant elements, $s$, and enumerated from the largest $s$ to $s = 1$. Some relevant applications would be Laroia and Farvardin’s construction of fixed rate quantizers from scalar quantizers [37] and Boncelet’s work with block arithmetic coding [36].

For an arbitrary distribution, the high rate channel error gain by conditional product enumeration over magnitude or linear enumeration is the ratio between the variance of the vector element probability distribution and the variance of the distribution of the absolute value of the vector element. In terms of symmetric distributions, let $X$ be a distribution that is strictly non-negative and has zero mass at $0$. Let $B$ be a random variable taking value of $1$ or $-1$ with probability $1/2$ each. If the source takes the distribution of $BX$, the gain of product code enumeration is

$$\text{Gain} = 10\log_{10}\left(\frac{\text{Var}(BX)}{\text{Var}(X)}\right) = 10\log_{10}\left(\frac{E[X^2]}{\text{Var}(X)}\right) \quad . \tag{40}$$

For a Laplacian (pyramidal) source, the gain is 3 dB; for a Gaussian (spherical) source, the gain is 4.3 dB.

## 5.5 Index Overflow Handling

Since PVQ codeword indices are transmitted at a fixed rate, and the number of codewords is not necessarily a power of two, some of the larger indices do not correspond to a valid codeword. For example, in the magnitude, linear, and conditional product enumeration, the number of unused indices is

$2^{\lceil \log N(l,k) \rceil} - N(l,k)$ (the conditional product-product enumeration can be handled similarly). These unused indices offer the potential for *error detection* at the decoder, whereupon corrective methods can be employed when an unused index is received. We investigate three corrective methods for index overflow here; see section 6.1 for simulations.

1) ***Zero*** method: Output the vector of all zeroes. This corresponds to the average value of the pyramid vector quantizer codebook. This is a simple technique, but not the best. Depending on the overflow index, the exact bit that flipped can sometimes be detected.

2) ***MSB*** method: Flip the most significant bit. For index overflow conditions, the most significant bit will always be set, and flipping the most significant bit always results in a valid index. In most practical pyramid enumerations, the most significant bit is most likely to be wrong since that is the most likely bit to cause index overflow. In many instances, the most significant bit is the only possible incorrect bit.

3) ***Even*** weight method: Set the output to the average of all possible legitimate vectors obtained by flipping a single set bit to zero in the received index $I_r$ that will result in a valid code. Clearly, this is the best solution in a mean-squared-error sense, but it is also the most complicated and hardware inefficient method of the three.

# 6 Simulations

The simulations on codebook performance are based on the effects of single bit errors on the PVQ codebook indices because it makes exhaustive testing feasible. For a pyramid codebook $P(l,k)$, whose code index is $n$ bits, and whose vector error due to a single bit index error is $E_{sb}$, we can derive the normalized single bit error (per pel) as $E_{nsb} = E_{sb}/(lk^2)$, which distributes error over the entire vector length, and compensates for different pyramid radii. For overall codebook error, we wish to penalize for long index lengths, $E_{ncb} = nE_{nsb}$. The expected PVQ error per pel for a unit-radius pyramid, under transmission of an index through a noisy channel with bit error rate, $p_{berr}$, is then $p_{berr}E_{ncb}$, as long as the assumption of low bit error rate and single bit error per index holds.

clarification, my mistake

## 6.1 Index Overflow Simulations

In this section, we compare the methods of handling index overflow discussed in section 5.5. We focus our attention on the case where the vector length is $l = 6$ and the codebook radius range is $1 \le k \le 18$. As shown in figure 6, the MSB correction method of handling overflows is superior to the ZERO method for product enumeration (and in general). Since MSB correction is simpler to implement than EVEN correction and generally better than ZERO correction, MSB correction is used in the following simulations to handle all index overflows.

## 6.2 Indexing Method Simulations

We have simulated the code vector error resulting from an index bit error per position in the index, both exactly and by theory for the MSB index overflow correction for the magnitude, linear, and product enumeration. The results are shown for $P(6, 18)$ in figure 7. The theoretical bound uses a somewhat more sophisticated model than equation (33) (see [41] for more details) that uses longer codeword length for the first vector elements than later vector elements; the difference results in enumeration error curves that are concave rather than perfectly linear (as expected through our simplified model described in this paper), but this second order effect is not large. The convergence of the theory and simulation curves on the right side of figure 7 is because we know that the last bit of the magnitude and product enumerated code is a sign bit, with error exactly given by equations (38) and (57). We conclude from figure 7 that the most significant bits are generally the best bits to protect, followed by the least significant bits, in magnitude and product enumeration.

Finally, we simulate the various enumeration methods with MSB index overflow correction. The five methods described are as follows: 1) Random enumeration using equation (32); 2) Fischer’s magnitude enumeration; 3) Linear enumeration; 4) Product enumeration; 5) Product-product enumeration. Exact simulation results over a wide range of radii, $1 \le k \le 60$, are only available for a smaller dimension, $l = 4$, because of the extremely large codebooks involved and simulation size constraints. As shown in figure 8, the radial codebook error due to single bit index error by product enumeration is reduced by about 3 dB over the previous enumeration methods, and by 6 dB over a purely random arranged codebook. The theoretical estimates are useful to estimate the channel error performance of these codebooks when the codebook size becomes too large for exact simulations.

# 7 Pyramid Vector Quantization for Images

Pyramid vector quantization has been previously used for DCT [7-14] and subband [15-24] image compression because both the subband and transformed image data are similar to the Laplacian in distribution. PVQ also has been considered in the context of matched joint source-channel coding for images [7][14][15], where the channel statistics are stationary and known. The following three sections demonstrate the advantages of the new error-resilient PVQ coding schemes introduced here for subband image compression under varying channel conditions.

The pyramid vector quantizer, by itself, assigns points to the nearest codeword on a single pyramid shell. For image compression applications where the vector lengths are small, the pyramid vector quantizer is typically matched with a radial quantizer to form several concentric pyramid shells [1]. This is called *polar* or *product* quantization, where the radius is quantized independently from the position on the shell. We describe this method as polar quantization, in order to avoid confusion with the enumerative technique.

Polar pyramid vector quantization utilizes four steps to code an input vector into digital bits. The first step is to find the nearest pyramid shell to the input vector. The second step is to scale the point onto the cubic lattice. The third step is to round the point to the nearest point in the cubic lattice that is on the pyramid shell. The fourth step is to enumerate that cubic lattice point, a process which assigns a unique index to each point on the pyramid shell. These steps are shown in figure 9.

In exhaustively examining the radial versus shell quantizer bit allocations, we have found that the radial position index should be allocated about as many bits as the typical sample element in the vector. This means an overall quantizer resolution of 7 bits per sample element implies that the radial quantizer resolution should be 7 bits. For very large vectors, the radial position index takes up a diminishing fraction of the overall bits; this can be attributed to the asymptotic equipartition principle – for very large vectors, most of the points cluster uniformly on very few shells.

## 8 Fixed rate PVQ subband coding

remove orthonormal

The subband filter experiments use a 9 tap quadrature mirror filter (QMF) with coefficients from the center tap of (0.5645751 0.2927051 -0.05224239 -0.04270508 0.01995484) [29]. This filter was chosen for its good overall performance, for its equal energy normalization into different frequency bands, and for its reasonable length. The filter is recursively applied to the input image, breaking it into four levels of subband decomposition as shown in figure 10 [29][30]. Our four level subband decomposition results in 31 frequency bands, rather than 13 for a logarithmic decomposition or 512 for a full four level decomposition. This offers a good tradeoff between performance, computational overhead, and side information for the subband variances.

After the subband decomposition, each frequency band is coded using a fixed rate quantizer. The lowest frequency band (DC band) is quantized with a Gaussian Lloyd-Max scalar quantizer. The upper bands are first scrambled into a random order to break up any correlation and then quantized with a polar PVQ quantizer. The polar PVQ quantizers consists of a scaled Erlang radial quantizer, where the scaling is determined to best fit generalized Gaussian $\gamma = 0.6$ data. This is combined with the PVQ shell quantizer, to obtain the best radial/shell bit allocations that lie on the distortion-rate convex hull. The bit allocation for the polar PVQ quantizer is determined through integer bit-allocation techniques [38][39] using experimentally obtained distortion per bit figures.

As shown in figure 11, on the USC database images Lena, Couple, LAX, and Mandrill, the performance of the PVQ algorithm exceeds that of JPEG with scaled suggested quantization matrix and custom Huffman tables.

# 9 Channel robustness

For our channel robustness experiments, we use a simple channel model. The encoder always transmits the same compressed image regardless of the actual channel because it does not know how many receivers are listening or the channel noise characteristics at each receiver. The receivers, on the other hand, have knowledge of its own individual channel noise conditions – both the average bit error rate and the presence of total signal loss in a deep fade. This model conveys some of the problems associated with mobile communication [32][33].

In our PVQ experiments, we only protect the scalar quantized DC band by repeating the two most significant bits of each index 3 times. This incurs negligible overhead – about 0.016 bpp, and requires just a simple majority decoder to correct bit errors in the DC band. All other bands are PVQ encoded and are *not* protected to show the inherent error resilience offered by the new product enumeration technique.

The difference in performance between product enumeration and magnitude enumeration on Lena at 0.5 bpp and LAX at 0.75 bpp is shown Figure 12. The high rate, low frequency bands experience the greatest benefit – about 3 dB, translating to an overall image gain of up to 1.5 dB over a wide range of bit error rates.

For comparison, we use the JPEG coder with resynchronization at every 6 macroblocks (JPEG-R); and the JPEG coder with resynchronization and (2,1,6) Viterbi decoding [34][35] (JPEG-R (2,1,6)VD). These JPEG implementations are compared with the product enumerated PVQ technique in figure 13 for Lena at 0.5 bpp. Clearly PVQ offers both better intrinsic coding performance and additional error resilience. Only with sophisticated channel coding can the performance of JPEG be made error resilient, but that significantly reduces the noiseless source coding performance.

A direct comparison of the product enumerated and magnitude enumerated PVQ for the Lena and LAX images are shown in figures 14 through 17. The images show a significant advantage of product enumeration over magnitude enumeration in the reduction of large artifacts in the decoded PVQ image.

# 10 Conclusion

Pyramid vector quantization is a form of vector quantization that does not require large codebook storage and that has very simple systematic encoding and decoding algorithms. In this paper, we have introduced a new method of PVQ enumeration called product enumeration, which reduces the susceptibility to channel noise by up to 3dB over existing methods for no additional coding overhead. These new product enumeration techniques have efficient hardware variants and can be applied to other fixed-rate symmetric quantizers as well. We have also shown effective index overflow strategies that handle detectable errors when the received index is outside the set of possible indices, a common problem with fixed-rate enumerated codes.

These new error resilient PVQ techniques, when combined with subband coding, achieves better compression performance than the variable rate JPEG image compression standard, and with much greater robustness as well. In fact, over wide ranges of varying channel noise, the error resilient PVQ techniques can improve both mean squared error and visual fidelity over previous techniques and channel-coded JPEG, and can sustain image quality under the presence of one percent channel error, without channel coding of the PVQ data.

missed a couple of formulas, so it's really 19

replace "Identities"

# Appendix A Nineteen Important Enumerative Formulas

The fundamental pyramid structure equations relate intimately to the number of points $N(l, k)$ in the pyramid codebook $P(l, k)$. In this chart, $l$ represents the vector length, $k$ represents the radius, $n$ represents the magnitude of a significant element, and $s$ represents the number of significant elements. Full details on their derivation and other asymptotic properties are described in [41].

$$N(l, k) = \frac{l}{k} N(k, l) \tag{41}$$

dimension-radius reciprocity

$$(l-1)N(l, k) = 2kN(l-1, k) + (l-1)N(l-2, k) \tag{42}$$

vector length recursion [14]

$$kN(l, k) = \boxed{2}lN(l, k-1) + (k-2)N(l, k-2) \tag{43}$$

should be "l"

vector radius recursion [14]

$$N(l, k) = \sum_s 2^s \binom{l}{s} \binom{k-1}{s-1} \tag{44}$$

enumeration formula by products

$$N(l, k) = \sum_i \binom{l}{i} \binom{(l-1)+(k-i)}{l-1} \tag{45}$$

enumeration formula by convolutions

$$2^l \binom{k-1}{l-1} \le N(l, k) \le 2^l \binom{l+k-1}{l-1} \tag{46}$$

asymptotic sandwich for k large

$$2^k \binom{l}{k} \le N(l, k) \le 2^k \binom{l+k-1}{k} \tag{47}$$

asymptotic sandwich for l large

$$N_V(l, K) = \sum_{k=0}^{K} N(l, k) = (N(l+1, K) + N(l, K))/2 \tag{48}$$

volume summation formula

$$E[n|s, l, k] = k/s \tag{49}$$

magnitude of significant element for known s

$$E[n^2|s, l, k] = \frac{k(2k-s+1)}{s(s+1)} \tag{50}$$

energy of significant element for known s

$$E[\|\mathbf{x}\|^2|s, l, k] = sE[n^2|s, l, k] \tag{51}$$

codebook energy for known s

$$E[\|\mathbf{x}\|^2|l, k] = \frac{k(k+1)}{l+1} \frac{(N(l, k+1) + N(l, k))}{N(l, k)} - k \tag{52}$$

codebook energy

$$E[n|l, k] = \frac{1}{2N(l, k)}(N(l+1, k) + N(l, k) - 2) \tag{53}$$

magnitude of significant element, n

$$E[n^2|l, k] = \frac{1}{2\boxed{N}(l, k)}\left(\frac{k+1}{l+1} 2N(l+1, k+1) - N(l+1, k) - N(l, k) - 4k - 2\right) \tag{54}$$

should be N

energy of significant element, n

$$E[s|l, k] = \frac{l}{N(l, k)}(N(l, k) - N(l-1, k)) \tag{55}$$

number of significant elements, s

$$E[\|\mathbf{x} - \mathbf{x}_S\|^2|l, k] \approx 2(E[\|\mathbf{x}\|^2|l, k] - kE[n|l, k]) \tag{56}$$

approximate error energy for shape error

$$E[\|\mathbf{x} - \mathbf{x}_B\|^2|l, k] = 4E[n^2|l, k] \tag{57}$$

error energy for sign error

$$E[\|\mathbf{x} - \mathbf{x}_S\|^2|s, l, k] \approx 2(E[\|\mathbf{x}\|^2|s, l, k] - k^2/s) \tag{58}$$

approximate error energy for shape error for known s

$$E[\|\mathbf{x} - \mathbf{x}_B\|^2|s, l, k] = 4E[n^2|s, l, k] \tag{59}$$

error energy for sign error for known s

missing formulas, my mistake.

## References

more info

[1] T. R. Fischer, “A pyramid vector quantizer,” *IEEE Transactions on Information Theory*, vol. 32, July 1986, pp. 568-583.

more info

[2] T. R. Fischer, “Geometric source coding and vector quantization,” *IEEE Transactions on Information Theory*, vol. 35, January 1989, pp.137-145.

[3] P. H. Westerink, J. H. Wever, D. E. Boekee, and J. W. Limpers, “Adaptive channel error protection of subband encoded images,” *IEEE Transactions on Communications*, vol. 41, no. 3, March 1993, pp. 454-459.

[4] F. Bellifemine, A. Capellino, A. Chimienti, R. Picco, and R. Ponti, “Statistical analysis of the 2D-DCT coefficients of the differential signal for images,” Signal Processing: *Image Communications*, 4 (1992), pp. 477-488.

[5] N. Tanabe and N. Farvardin, “Subband image coding using entropy-coded quantization over noisy channels,” *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 5, June 1992, pp. 926-943.

[6] R. C. Reininger and J. D. Gibson, “Distributions of the two-dimensional DCT coefficients for images,” *IEEE Transactions on Communications*, vol. 31, no. 6, June 1983, pp. 835-839.

[7] M. J. Ruf and P. Filip, “Combined source and channel coding for transmission of PVQ compressed images over Gaussian and Rayleigh fading channels,” *Proceedings of the Picture Coding Symposium ‘93*, Lausanne Switzerland.

[8] B.-L. Yeo, M. M. Yeung, S. Sarkar, “A fixed-rate vector quantizer based on pyramid-bounded integer lattices for image compression,” *IEEE International Conference on Image Processing 1994*, pp. 578-582.

[9] T. R. Fischer, “Entropy-constrained geometric vector quantization for transform image coding,” *IEEE ICASSP 1991*, Toronto Canada, pp. M2269-M2272.

[10] H.C. Tseng and T. R. Fischer, “Transform and hybrid transform/DPCM coding of images using pyramid vector quantization,” *IEEE Transactions on Communications*, vol. COM-35, no. 1 January 1987, pp. 79-86.

[11] M. E. Blain and T. R. Fischer, “Optimal rate allocation in pyramid vector quantizer transform coding of imagery,” *IEEE ICASSP ‘87*, pp. 729-732.

[12] P. D’Alessandro, P. Formenti, and R. Lancini, “Pyramid vector quantization applied to a video coding scheme for contribution quality,” *IEEE ICASSP ‘93*, pp. V-249-V252.

[13] P. D’Alessandro and R. Lancini, “Video coding scheme using DCT-pyramid vector quantization,” *IEEE Transactions on Image Processing*, vol. 4, no. 3, pp. 309-319, March 1995.

more info

[14] P. Filip and M.J. Ruf, “A fixed-rate product pyramid vector quantization using a bayesian model,” *IEEE Globecom* 1992.

[15] M. J. Ruf, “A high performance fixed rate compression scheme for still image transmission,” *Data Compression Conference ‘94*, Snowbird Utah, pp. 294-303.

upper case “E”

[16] A. C. Hung and T. H.-Y. Meng, “Error resilient pyramid vector quantization for image processing,” *IEEE International Conference on Image Processing 1994*, pp. 583-587.

[17] E. K. Tsern and T. H.-Y. Meng, “Pyramid vector quantization for image compression,” *IEEE ICASSP 1994*, April 1994, pp. V601-V604.

[18] E. K. Tsern, A. C. Hung, and T. H.-Y. Meng, “Video compression for portable communication using pyramid vector quantization of subband coefficients, *Proceedings 1993 IEEE Workshop on VLSI Signal Processing*, Koningshof Netherlands, October 1993, pp. 444-451.

[19] T. H. Meng, B. M. Gordon, E. K. Tsern, A. C. Hung, “Portable video-on-demand in wireless communication,” *Proceedings of the IEEE*, vol. 83, no. 4, April 1995, pp. 659-680.

[20] E. K. Tsern and T. H.-Y. Meng, “A low power, video-rate pyramid VQ decoder,” *ISSCC Digest of Technical Papers,* February 1996, pp. 162-163.

[21] D.G. Jeong and J.D. Gibson, “Image coding with uniform and piecewise-uniform vector quantizers,” *IEEE Trans. Image Processing,* February 1995, pp. 140-146.

[22] Z. Mohd-Yusof and T. R. Fischer, “An entropy-coded lattice vector quantizer for transform and subband image coding,” *IEEE Trans. Image Processing,* February 1996, pp. 289-298.

[23] M. Barlaud, P. Sole, M. Antonini, P. Mathieu, T. Gaidon, “Pyramidal lattice vector quantization for multiscale image coding,” *Universite de Nice, Sophia Antipolis*, Report 91-26.

[24] M. Antonini, M. Barlaud, P. Mathieu, “Image coding using lattice vector quantization of wavelet coefficients,” *IEEE ICASSP-92*, pp. IV-401-404.

[25] M. Berger, *Geometry I*, Springer Verlag, New York, 1994.

[26] M. Berger, *Geometry II*, Springer Verlag, New York, 1977.

[27] H. S. M. Coxeter, *Regular Polytopes*, Dover Publications, New York, 1973.

[28] P. F. Swaszek, “A vector quantizer for the Laplace source,” *IEEE Transactions on Information Theory*, vol. 37, No. 5, Sept. 1991, pp. 1355-1365.

[29] T. Senoo and B. Girod, “Vector quantization for entropy coding of image subbands,” *IEEE Transactions on Image Processing*, vol. 1, no. 4, October 1992, pp. 526-532.

[30] S. G. Mallat, “A theory for multiresolution signal decomposition: the wavelet representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, July 1989, pp. 674-693.

[31] M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies, “Image coding using the wavelet transform,” *IEEE Trans. on Image Processing,* vol. 1, no. 2, April 1992, pp. 205-220.

[32] D. L. Cox, “Universal digital portable radio communications,” *Proceedings of the IEEE*, vol. 75, no. 4, April 1987, pp. 436-477.

[33] W. C.-Y. Lee, *Mobile Cellular Telecommunications Systems*, McGraw Hill, New York, 1989.

[34] G. C. Clark Jr. and J. B. Cain, *Error-Correction Coding for Digital Communications*, Plenum Press, New York, 1981.

[35] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, 1983.

[36] C. G. Boncelet, Jr. “Block arithmetic coding for source compression,” *IEEE Transactions on Information Theory*, vol. 39, no. 5, September 1993, pp. 1546-1554.

[37] R. Laroia and N. Farvardin, “A structured fixed-rate vector quantizer derived from a variable-length scalar quantizer, Part I - memoryless sources,” *IEEE Transactions on Information Theory,* vol. 39, no. 3, pp. 851-867.

[38] A. K. Jain, “Image data compression, a review,” *Proceedings of the IEEE*, vol. 69, no. 3, March 1981, pp. 349-389.

[39] A. Segall, “Bit allocation and encoding for vector sources,” *IEEE Transactions on Information Theory,* vol. 22, no. 2, March 1976, pp. 162-169.

[40] A. C. Hung and T. H.-Y. Meng, “Adaptive channel optimization of vector quantized data,” *Data Compression Conference 1993*, Snowbird, Utah, pp. 282-291.

[41] A. C. Hung, *Geometric Coding for Error Resilient Image Compression*, Ph.D. thesis, Stanford University, 1994.

***** *** NOT PART OF THE MAIN TEXT *** *****

### List of figures