Block-Adaptive Lattice Vector Quantization in Image Coding

Wisarn Patchoo\textsuperscript{1} and Thomas R. Fischer\textsuperscript{2}, Non-members

ABSTRACT
A subband image coding algorithm is described based on lattice-based spherical VQ and lattice-based pyramid VQ. The algorithm partitions a subband (or wavelet) decomposed image into blocks of various sizes, depending on their energy and complexity constraints on the enumeration encoding of lattice codevectors. Each block is lattice vector quantized and encoded using a product code. The algorithm is simple and effective, exploiting energy clustering in a wavelet transformed image. Using the integer lattice, the algorithm provides performance slightly better than the Set-Partitioning Embedded Block (SPECK) algorithm, and is competitive with JPEG2000 and the set partitioning in hierarchical trees (SPIHT) algorithm.

Keywords: Lattice Vector Quantization, Image Coding, and Wavelet Transform

1. INTRODUCTION
Subband image decomposition [1], typically using a wavelet transform [2], is an effective method of image representation, supporting several desirable image compression properties, such as scalability, region-of-interest coding, and embedded coding for progressive transmission [3]-[6]. An octave subband structure is commonly used (e.g., [5]-[6]). In all except the lowest frequency subband, the subband (or, referred to interchangeably in this paper, wavelet) coefficients have marginal density that is well-modeled as Laplacian, or generalized Gaussian (GG) with small shape parameter ($\alpha$), i.e., $\alpha = 0.7$, as demonstrated in Fig. 1 [2], [7]. The subband coefficients also have small correlation [7], but they are generally not memoryless, instead displaying an evident nonlinear energy dependence. This is shown in Fig. 2, comparing the empirical block energy density for contiguous blocks of subband coefficients, to the energy density of blocks of memoryless Laplacian or generalized Gaussian data of the same mean and variance. The empirical block energy density is highly peaked at small energy, and heavy-tailed, reflective of the large number of small amplitude subband coefficients, with the larger magnitude coefficients typically clustered, rather than distributed uniformly throughout the subbands. Fig. 2 uses block energy defined as $|X|^2 = \sum_{i \in \text{block}} |x_i|^2$, but similar results are obtained defining the block energy as the $\ell_1$ norm. Efficient subband image coding algorithms take advantage of such energy dependence, such as by using a quad-tree data structure across [3]-[4] or within subbands [8], or by using local contexts in bit-plane coding [5].

The image subband block energy density can be modeled as a mixture of memoryless Gaussian or Laplacian random variables [9]-[10]. A suitable block-based lattice vector quantizer (LVQ) can be formed using concentric spherical (motivated by the Gaussian mixture model), or pyramidal LVQ (motivated by the Laplacian mixture model). A spherical LVQ using the RE\textsubscript{8} lattice [11] is used in the AMR-WB+ audio coding standard [12]. A pyramid-based LVQ has been used for image coding [13]-[14]. Lattice-based vector quantization offers the potential of relatively simple quantization and granular coding gain [15]. However, the mapping from lattice codevectors to binary codewords can be cumbersome, with complexity that typically grows with the lattice dimension.

In this paper, we propose an image coding algorithm based on lattice-based spherical vector quantization (LSVQ) or lattice-based pyramid VQ (LPVQ), motivated by Fig. 2. Quadtree partitioning is used to divide regions of coefficients into blocks to be lattice vector quantized and encoded. The block sizes are
The quantization and encoding can be performed efficiently. In this work, partitioning based on quadtrees, similar to [16], is used and the testing condition uses either the $\ell_2$ or $\ell_1$ norm for set energy. More specifically, an $\ell$-limited set in this work implies a set that can be quantized and encoded using a product code based on LSVQ or LPVQ and hence, the vector of set coefficients lies within a bounding sphere or pyramid of respective $\ell_2$ or $\ell_1$ radius.

Let $c_{i,j}$ and $\hat{c}_{i,j}$ be the unquantized and corresponding quantized version of the wavelet coefficient at coordinate $(i,j)$ of a transformed image of size $N \times N$. Denote a subset (or block) at level $l$ due to quadtree partitioning, and its corresponding quantized version, as $S_{m,n}^{(l)}$ and $\hat{S}_{m,n}^{(l)}$, respectively, where $(m,n)$ is the coordinate of a given coefficient in $S_{m,n}^{(l)}$ used to refer to that subset, e.g., the coordinate of the coefficient at the block top-left corner, as shown in Fig. 3. Denote subsets (or subblocks) resulting from quadtree partitioning $S_{m,n}$ as $O(S_{m,n}^{(l)})$. Define $E(S_{m,n}^{(l)}, \nu) = \sum_{\nu \in S_{m,n}^{(l)}} |c_{i,j}|^\nu$, where $\nu \in \{1, 2\}$, as the energy in a block of quantized coefficients. A set $S_{m,n}^{(l)}$ is said to be $\ell$-limited if

$$E(S_{m,n}^{(l)}, \nu) \leq T^{(l)}$$

(1)

where $T^{(l)}$ is a predefined threshold for partition level $l$, and the threshold can vary with partition level.

The LSVQ and LPVQ use as codevectors all lattice vectors within a bounding sphere or pyramid as shown in Fig. 4 and Fig. 5, respectively. Let $n$ denote the vector dimension and $\Lambda_n$ the lattice. We assume the lattice is of the form described in [19], with $\Lambda_n \subseteq \mathbb{Z}^n$, where $\mathbb{Z}^n$ is the cubic lattice of dimension $n$. Let $N_{\Lambda,1}(n,k)$ denote the number of lattice points satisfying $||y||_1 = k$, and let $N_{\Lambda,2}(n,k)$ denote the number of lattice points satisfying $||y||_2 = k$, $k = 0, 1, \ldots$. These values are summarized as the lattice nu-function [13] and theta function [15], respectively. A variable-length code, $C_\nu$, (e.g., a Huffman code) is used to encode the $\nu$-norm energy, with codeword denoted by $c_\nu(||y||_\nu)$. An enumeration code, $C_\nu$, uses $\nu \log_2(N_{\Lambda,\nu}(n, ||y||_\nu))$ bits/vector to encode the codevector, conditioned on the energy. Thus, the encoding bits for the quantized block $\hat{S}_{m,n}^{(l)}$ are composed of two parts: 1) a codeword used to specify the sphere or the pyramid, $c_\nu(||y||_\nu)$, and 2) a codeword
used to specify the codevector on a given sphere or pyramid, $c_c$. Encoding and decoding using codes $C_C$ and $C_c$ can be implemented primarily using look-up tables.

Fig. 6(a) shows the relative distribution of blocks of various sizes resulting from the quadtree partitioning based on the proposed algorithm and the $\ell_2$ energy measure. Similar distributions are obtained for the $\ell_1$ case as shown in Fig. 6(b). At low overall encoding rate there are many large blocks and very few singleton blocks, most of the latter from the low frequency subband. Even at large encoding rate, there are relatively few singleton blocks, and the abundance of $2 \times 2$ and $4 \times 4$ blocks suggest the granular gains of the $D_4$, $E_8$, 16-dimensional Barnes-Wall lattice ($\Lambda_{16}$), or other lattices might improve the encoding signal-to-noise ratio (SNR).

3. CODING ALGORITHM

3.1 Algorithm

The main idea of the partitioning algorithm is simple. Large subsets are tested according to an energy threshold to determine whether partitioning is required. As soon as subsets have energy within the testing threshold, they are encoded using a product code based on LSVQ ($\nu = 2$) or LPVQ ($\nu = 1$) and are removed from further consideration. Since the order in which subsets are tested is important and needs to be maintained, a list is used to store the testing order. Denote the list of sets as $LS$. Also, define the binary testing function as

$$EF(\hat{S}_{m,n}^{(l)}, \nu) = \begin{cases} 0, & \text{if } E(\hat{S}_{m,n}^{(l)}, \nu) \leq T^{(l)} \\ 1, & \text{if } E(\hat{S}_{m,n}^{(l)}, \nu) > T^{(l)}. \end{cases}$$

Assume $N$ is power of 2. The proposed encoding algorithm is presented in Algorithm 1.

The algorithm uses the functions $LVQ()$ to perform lattice VQ, $encode()$ to generate the binary codeword representing a lattice codevector, and $encode_{scalar}()$ to generate the binary codeword for a singleton block. To quantize the block $S_{m,n}^{(l)}$ using LVQ, the vector is formed based on raster scanning. The decoding algorithm is similar to the encoding algorithm, replacing $output$ to $input$, $encode()$ to $decode()$, and quantization is no longer required. In the algorithm main loop (while loop), the block size is decreased if the block is non-e-limited. The decoder tracks the block size from the sequence of energy-testing bits. The algorithm makes no assumption on the lattice (or lattices) used. However, if the integer lattice is used, lattice quantization can be done
Algorithm 1 Block-Adaptive LVQ

Given: subband image \( \{c_{ij}\} \) and chosen \( \nu \)-norm

Initialize: \( LS = \{(0, 0)\} \), \( l = 0 \)

Start:

1. \textbf{while} \( LS \neq \phi \) \textbf{do}
2. \hspace{1em} \textbf{for all} \( (m, n) \in LS \) except those added in current \( l \) \textbf{do}
3. \hspace{2em} \textbf{if} \( N > 1 \) \textbf{then}
4. \hspace{3em} \( \hat{S}_{m,n}^{(l)} = LVQ(S_{m,n}^{(l)}, N) \)
5. \hspace{3em} \textbf{output} \( EF(\hat{S}_{m,n}^{(l)}, \nu) \)
6. \hspace{3em} \textbf{if} \( (EF(\hat{S}_{m,n}^{(l)}, \nu) = 1) \) \textbf{then}
7. \hspace{4em} \textbf{add} \( O(S_{m,n}^{(l)}) \) to the end of \( LS \)
8. \hspace{4em} \textbf{remove} \( (m, n) \) from \( LS \)
9. \hspace{3em} \textbf{else}
10. \hspace{4em} \textbf{output} \( encode(\hat{S}_{m,n}^{(l)}) \)
11. \hspace{4em} \textbf{remove} \( (m, n) \) from \( LS \)
12. \hspace{3em} \textbf{end if}
13. \hspace{3em} \textbf{else}
14. \hspace{4em} \( \hat{S}_{m,n}^{(l)} = \text{round}(S_{m,n}^{(l)}) \)
15. \hspace{4em} \textbf{output} \( encode_{\text{scalar}}(S_{m,n}^{(l)}) \)
16. \hspace{4em} \textbf{remove} \( (m, n) \) from \( LS \)
17. \hspace{3em} \textbf{end if}
18. \hspace{2em} \textbf{end for}
19. \hspace{1em} \( N \leftarrow N \gg 2 \)
20. \hspace{1em} \( l \leftarrow l + 1 \)
21. \textbf{end while}

in advance, before the algorithm execution, and the partitioning can be done directly to quantized blocks, \( \hat{S}_{m,n}^{(l)} \).

3.2 Complexity consideration

In addition to the complexity of the wavelet transform, which also used in the JPEG2000 and SPIHT algorithms, the block-adaptive LVQ algorithm complexity has three main parts: 1) lattice VQ and encoding, 2) subblock energy testing and encoding, and 3) memory requirement. The lattice VQ is composed of three steps: vector quantization, the \( \nu \)-norm energy encoding, and the enumeration encoding. Vector quantization is very simple when the \( Z_n \) lattice is used (just integer rounding operation), however, it can be complicated when other lattices are used. We give further explanation of this in Section 4. The \( \nu \)-norm energy encoding can be done easily using a look-up table. The enumeration encoding can be cumbersome if the lattice dimension (equivalent to size of image subblock) becomes large, since the number of lattice points needed to be enumerated increases exponentially with the dimension or rate. However, since the allowable subblock sizes are predefined, and the block energy is limited, the encoding can be simply performed using looking-up tables. It should be remarked that in the proposed algorithm the lattice VQ is performed at every partitioning level so that even a simple look-up table operation may require some running time. However, when the \( Z_8 \) lattice is used, the lattice VQ need only be performed once before the partitioning is done. This helps reducing encoding time. Since the SPIHT algorithm can be considered as implicitly using entropy coded scalar quantization, the lattice VQ in the proposed algorithm has additional complexity unless the \( Z_n \) lattice is used. However, for certain lattices, such as the \( D_n \) and \( E_8 \) lattices, this additional complexity is very modest [15].

For subblock energy testing, during the partitioning each subblock energy is tested against predefined energy thresholds. Since a large subblock is composed of a group of smaller subblocks (corresponding to the quadtree partitioning), and the a large block's energy is the sum of its subblock energies, a linked list data structure can be used and the subblock energy calculation performed only once before coding the encoding algorithm is executed. During the coding, a specified subblock’s energy can be obtained from that list.

Finally, the algorithm’s memory requirements are similar to the SPIHT algorithm. In addition to the wavelet transformed image, the partitioning requires the implementation of a linked list, as well as the list of subblock energies. Also, the proposed algorithm requires memory for storing the enumeration look-up tables, which may be large if large allowable block sizes or energy thresholds are used.

4. SIMULATION RESULTS

To evaluate the performance of the proposed algorithm, a five-level subband decomposition is generated using a 9/7 biorthogonal filter bank [4], [5]. At first, the simulation uses integer (cubic) lattice VQ for simplicity. Later on, the simulation using other lattices will be investigated. Both LVQ based on \( \ell_1 \) (LPVQ) and \( \ell_2 \) norms (LSVQ) are used. A Huffman code is generated for \( C_n \), using empirical energy probabilities for each partition level. The largest allowed block size is \( 16 \times 16 \) \( (N = 16) \) so that the allowed possible block sizes due to partitioning are \( 16 \times 16, 8 \times 8, 4 \times 4, 2 \times 2, \) and \( 1 \times 1 \). Raw energy-testing bits are entropy coded (using arithmetic coding) to exploit the statistical dependence among them. Note that the 9/7 biorthogonal filter is used in the simulation since it is commonly used in most image coding algorithm literatures so that the performance comparison with other image compression algorithms can be done directly. Moreover, by the study using impulse and step response criteria, and using Holder regularity in [18] shows that at the same compression ratio, the 9/7 biorthogonal filter bank offers highest PSNR among all of the minimum order biorthogonal filter banks with combined analysis/synthesis filter lengths of \( \leq 36 \). In general, any other perfect reconstruction filter bank can be used together with the proposed
algorithm.

The simulation results are shown in Fig. 7-9 compared with the SPIHT and the SPECK algorithm (SPECK is done using QccPack [26]). The performance is slightly superior to SPECK, and is competitive with the SPIHT algorithm (using adaptive arithmetic coding), with LPVQ performing a little better at low and moderate encoding rates. The LPVQ performs slightly better than the LSVQ. Note that the total encoding rate would increase about 3-5% without entropy coding of the energy-testing bits. So, there is a little gain by entropy coding those bits. About 5-10% of the total encoding rate is used to encode the energy-testing bits.

Figure 6 suggests the possibility that better covering lattices (higher granular gain compared with the $Z$ lattice) might be used, based on the partitioned block sizes, to possibly allow improvement in LVQ SNR due to granular fidelity. Since there are four possible allowed block sizes, $16 \times 16$, $8 \times 8$, $4 \times 4$, and $2 \times 2$, the vectors formed by these blocks are of dimension 256, 64, 16, and 4, respectively. Then, we might be able to use better lattices with dimensions corresponding to those vector dimensions. However, for vectors of size 256 or 64, using other better lattices rather than integer lattice may require very large computational complexity. To see this, let us briefly describe the notion of the coset codes introduced by Forney [19]-[20].

Forney shows that a family of Barnes-Wall lattices and its principal sublattices can be generated by iteratively applying a simple construction called the squaring construction. He also points out the close relationship between this family of lattices and the family of Reed-Muller codes (RM) [21]. More specifically, denote the lattice generated by the iterated squaring construction as $\Lambda(r, n)$, where $r$ is an integer and the lattice dimension is $N = 2^{n+1}$. $\Lambda(r, n) = Z^N$ for $r \geq n$, and $\Lambda(r, n) = R^{-r}\Lambda(0, n)$ for $r \leq 0$, where $R$ is the rotation operator. The Barnes-Wall lattices are equivalent to $\Lambda(r, n)$ when $r \geq 0$. Forney shows that $\Lambda(r, n)$ can be constructed as the union of the $2^{m/2}Z^{2N}$ lattice translated by codewords of Reed-Muller codes, where $m = n - r$ when $n - r$ is even, and $m = n - r + 1$ when $n - r$ is odd. For example, using Forney’s expression, the Gosset ($E_8$) lattice can be written as

$$E_8 = \Lambda(0, 2) = 2Z^8 + (8, 4, 4),$$

where the sum is a direct sum and $(8, 4, 4)$ is the $(n, k, d)$ Reed-Muller code with block length $n$, $k$ information bits, and minimum Hamming distance $d$ [21]. The above expression shows that the $E_8$ lattice is obtained as the union of $Z^8$ lattice scaling by a factor 2 and translated by each codeword of the $(8, 4, 4)$ RM code. It also tells us that performing LVQ using the $E_8$ lattice is equivalent to performing LVQ using the $2Z^8$ lattice plus its 15 cosets (the $2Z^8$ lattice is itself the zero coset).

Now, consider the following choice of lattices: 256-dimensional Barnes-Wall lattice, $\Lambda_{256}$, for vectors of size 256, and 64-dimensional Barnes-Wall lattice, $\Lambda_{64}$ for vector dimension 64 [15]. Using Forney’s expres-
sion, $\Lambda_{64}$ can be represented as
\[
\Lambda_{64} = \Lambda(0, 5) = 8Z^{64} + (64, 7, 82) + 2(64, 42, 8) + 4(64, 63, 2),
\]
where $(64, 7, 82), (64, 42, 8)$, and $(64, 63, 2)$ are Reed-Muller codes and the sum is a direct sum. Eq. (3) suggests that LVQ using $\Lambda_{64}$ is equivalent to LVQ using the $8Z^{64}$ (the 64-dimensional integer lattice scaled by the factor 8) combined with additional similar vector quantizations using its $2^7 + 2^{42} + 2^{63}$ cosets. For $\Lambda_{256}$, the number of cosets is far larger than this. Due to the complexity of using the 256- or 64-dimensional Barnes-Wall lattices for VQ, we instead use the simple integer lattice for these dimensions. The 16-dimensional Barnes-Wall lattice, $\Lambda_{16}$ and $D_4$ lattice [15] are used to quantize 16-dimensional and 4-dimensional vectors, respectively. An effective quantization algorithm for the $D_n$ lattice is described in [15]. The $\Lambda_{16}$ lattice can be expressed as [20]
\[
\Lambda_{16} = 2D_{16} + (16, 5, 8).
\]
The above expression suggests a simple way to perform LVQ using $\Lambda_{16}$. Let $c_i$, for $i = 1, \ldots, 32$, be the binary codewords of the (16,5,8) RM code. Let’s denote the 16-dimensional input vector formed by an image block of size 4 x 4 as $x$ and the lattice codevector as $y$. Then, LVQ using $\Lambda_{16}$ can simply be done as follows.

0) Let $y = VQ_{2D16}(x)$ denote lattice VQ using the $2D_{16}$ lattice.
1) For $i = 1$ to 32, form $y_i = c_i + VQ_{2D16}(x - c_i)$.
2) Find $i^* = \arg\min_i ||y - y_i||^2$.
3) The $\Lambda_{16}$ codevector closest to $x$ is $y_{i^*}$.

Since the proposed algorithm with LPVQ outperformed LSVQ in Section 4, the subsequent simulation results are based on LPVQ. Table 1 shows the simulation results of the proposed algorithm using only integer lattice, and that using the integer lattice together with the $D_4$ and $\Lambda_{16}$ lattices for 4-dimensional and 16-dimensional vectors, respectively. To compare with the results when only the integer lattice is used, the $D_4$ and $\Lambda_{16}$ lattices are scaled by the factors $1/1.1892$ and $1/1.6818$, respectively, to have the same point density as the integer lattice.

From the table, it is perhaps surprising that the simulation results using the integer lattice together with $D_4$ and $\Lambda_{16}$ lattices (LPVQ$_{Lat}$) provides lower PSNR than using only the integer lattice (LPVQ$_Z$) at the same encoding rate. However, this is consistent with the work of Gao, Belzer, and Villasenor [22], that shows that at low encoding rate, LVQ using the Z-lattice provides performance superior to LVQ using other better covering lattices such as the $E_8$ or Leech lattices. This follows from the observation that wavelet coefficients can be modeled well using a generalized Gaussian distribution [23] with low shape parameter (also shown in Fig. 1) and the Voronoi regions of the Z-lattice are better suited with the anisotropy of the generalized Gaussian density which has more vectors along the coordinate axes than off-axis. This effect influences significantly the performance of LVQ at low encoding rates. For high encoding rates, LVQ using lattice with higher granular gain still provides better performance. For our simulations with the Lena image, this occurs at encoding rates higher than about 2 bpp, and LPVQ$_{Lat}$ begins providing better performance over LPVQ$_Z$. It should be emphasized that this particular effect occurs with input source of low-shape-parameter generalized Gaussian distribution, but not of Gaussian distribution.

<table>
<thead>
<tr>
<th>Image</th>
<th>Rate (bpp.)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>0.125</td>
<td>31.13</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>34.05</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>37.15</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>40.54</td>
</tr>
<tr>
<td>Barbara</td>
<td>0.125</td>
<td>25.29</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>28.05</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>31.59</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>36.53</td>
</tr>
<tr>
<td>Goldhill</td>
<td>0.125</td>
<td>28.48</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>30.54</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>33.11</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>36.68</td>
</tr>
<tr>
<td>Aerial</td>
<td>0.125</td>
<td>23.54</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>25.86</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>28.83</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>33.14</td>
</tr>
<tr>
<td>Aerial2</td>
<td>0.125</td>
<td>26.60</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>28.54</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>30.65</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>33.45</td>
</tr>
</tbody>
</table>

LPVQ$_Z$: LPVQ using Z-lattice.
LPVQ$_{Lat}$: LPVQ using $Z$ together with $D_4$ and $\Lambda_{16}$ lattices.

$D_4$ has point density half of that of $Z^4$. This corresponds to a rate difference of 1 bit per 4 dimensions, or 0.25 bits per dimension. It implies that by scaling $D_4$ lattice with the factor $g$, denote scaled $D_4$ as $D_4^g$, the rate of $D_4^g$ must be higher than unscaled $D_4$ by 0.25 bit. Using this and by the fact that at high rate, the encoding rate can be approximately as $0.5 \log_2(4^{2/2}/D)$, we get $g^2 = 1/(2^{0.5})$, or $g = 1/(2^{0.5}) = 1/1.1892$.

Similar to $D_4$, point density of $\Lambda_{16}$ is $1/(2^{12})$ of that of $Z^{16}$. This corresponds to a rate difference of 12 bits per 16 dimensions, or 0.75 bits per dimension. Then, we get $g^2 = 1/(2^{0.75})$, or $g = 1/(2^{0.75}) = 1/1.6818$. 

---

1. $D_4$ has point density half of that of $Z^4$. This corresponds to a rate difference of 1 bit per 4 dimensions, or 0.25 bits per dimension. It implies that by scaling $D_4$ lattice with the factor $g$, denote scaled $D_4$ as $D_4^g$, the rate of $D_4^g$ must be higher than unscaled $D_4$ by 0.25 bit. Using this and by the fact that at high rate, the encoding rate can be approximately as $0.5 \log_2(4^{2/2}/D)$, we get $g^2 = 1/(2^{0.5})$, or $g = 1/(2^{0.5}) = 1/1.1892$.
2. Similar to $D_4$, point density of $\Lambda_{16}$ is $1/(2^{12})$ of that of $Z^{16}$. This corresponds to a rate difference of 12 bits per 16 dimensions, or 0.75 bits per dimension. Then, we get $g^2 = 1/(2^{0.75})$, or $g = 1/(2^{0.75}) = 1/1.6818$. 

---

Table 1: Performance comparison of proposed method with LPVQ using integer lattice, and $D_4$ and $\Lambda_{16}$ lattices

<table>
<thead>
<tr>
<th>Image</th>
<th>Rate (bpp.)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>0.125</td>
<td>31.13</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>34.05</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>37.15</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>40.54</td>
</tr>
<tr>
<td>Barbara</td>
<td>0.125</td>
<td>25.29</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>28.05</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>31.59</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>36.53</td>
</tr>
<tr>
<td>Goldhill</td>
<td>0.125</td>
<td>28.48</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>30.54</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>33.11</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>36.68</td>
</tr>
<tr>
<td>Aerial</td>
<td>0.125</td>
<td>23.54</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>25.86</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>28.83</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>33.14</td>
</tr>
<tr>
<td>Aerial2</td>
<td>0.125</td>
<td>26.60</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>28.54</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>30.65</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>33.45</td>
</tr>
</tbody>
</table>

LPVQ$_Z$: LPVQ using Z-lattice.
LPVQ$_{Lat}$: LPVQ using $Z$ together with $D_4$ and $\Lambda_{16}$ lattices.
Table 2: Performance comparison for several images

<table>
<thead>
<tr>
<th>Coding Method</th>
<th>Lena (512 x 512)</th>
<th>Barbara (512 x 512)</th>
<th>Goldhill (512 x 512)</th>
<th>Aerial (512 x 512)</th>
<th>Aerial2 (2048 x 2048)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPEG2000</td>
<td>PSNR (dB)</td>
<td>PSNR (dB)</td>
<td>PSNR (dB)</td>
<td>PSNR (dB)</td>
<td>PSNR (dB)</td>
</tr>
<tr>
<td>0.0625 bpp.</td>
<td>28.05</td>
<td>23.38</td>
<td>26.56</td>
<td>21.27</td>
<td>24.63</td>
</tr>
<tr>
<td>0.125 bpp.</td>
<td>31.22</td>
<td>25.55</td>
<td>30.46</td>
<td>23.10</td>
<td>26.51</td>
</tr>
<tr>
<td>0.25 bpp.</td>
<td>34.28</td>
<td>28.55</td>
<td>30.46</td>
<td>25.42</td>
<td>28.58</td>
</tr>
<tr>
<td>0.5 bpp.</td>
<td>37.43</td>
<td>32.48</td>
<td>33.18</td>
<td>28.71</td>
<td>30.64</td>
</tr>
<tr>
<td>1.0 bpp.</td>
<td>40.61</td>
<td>37.37</td>
<td>36.58</td>
<td>33.19</td>
<td>33.28</td>
</tr>
<tr>
<td>2.0 bpp.</td>
<td>44.84</td>
<td>43.17</td>
<td>41.93</td>
<td>39.82</td>
<td>38.14</td>
</tr>
</tbody>
</table>

The LPVQ\textsubscript{Z} encoding performance is compared to other well-known coding methods. The comparison is shown in Table 2. The JPEG2000 encoding is done using OpenJPEG [25] and the SPECK encoding is done using QccPack [26]. From Table 2, it can be seen that the performance of LPVQ\textsubscript{Z} is competitive with JPEG2000 and SPIHT-AC, but is slightly better than SPECK. At low encoding rate (\(<0.5\) bpp), LPVQ\textsubscript{Z} slightly outperforms the other encoding methods, except JPEG2000. For some images such Goldhill and Aerial and at very low encoding rates, LPVQ\textsubscript{Z} provides PSNR increase of about 0.2-0.4 dB. This suggests that the proposed method is particularly effective at low encoding rates.

5. CONCLUSION

In this paper, we propose an algorithm based on LPVQ and LSVQ for image coding. The proposed algorithm partitions the subband (or wavelet) image into blocks of various sizes by comparing $\ell_1$ or $\ell_2$ block energy, respectively, with the thresholds defined according to complexity constraints on enumeration encoding of lattice points. We show that based on the proposed algorithm the energy clustering can be exploited effectively. From the simulations, the proposed algorithm with LPVQ provides a performance little better than SPECK algorithm, and competitive with JPEG2000 and SPIHT using arithmetic coding. The proposed algorithm works very well at low encoding rates. Coding blocks or sets of coefficients using LVQ also allows the possible improvement in SNR by using better covering lattices, such as $D_4$ or $E_8$ or $\Lambda_{16}$. However, the above results show that LVQ using better covering lattices to encoding wavelet coefficients gives higher PSNR over LVQ using Z-lattice only at higher encoding rates. It should be noted that for some images the granular fidelity of lattices may possibly provide better visual quality of decoded image compared with SPIHT [27]. Future work will emphasize modifying the LSVQ and LPVQ encoding to provide an embedded bitstream which will allow progressive transmission of images.

References


Wisarn Patchano received Bachelor degree from Kasetsart University in 2000 and Master degree from King Mongkut’s Institute of Technology Ladkrabang in 2003. Both are in Electrical Engineering. From 2002 to 2005, he was a Intel- ligent Network Engineer at the Huawei Technologies (Thailand). From 2006 until now, he is with School of Engineering, Bangkok University, Thailand. He received Ph.D in Electrical and Computer Engineering from Washington State University in 2011. His research interests are data compression, signal and image processing.

Thomas R. Fischer received the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Massachusetts, Amherst, and the Sc.B. degree magna cum laude from Brown University. From June 1975 until August 1976, he was a Staff Engineer at the Charles Stark Draper Laboratory, Cambridge, MA. From 1979 until 1988 he was with the Department of Electrical Engineering. Since January 1989 he has been a Professor in the School of Electrical Engineering and Computer Science at Washington State University. From 1999 until 2004 he was Director of the School of EECS. His current research interests include data compression, image and video coding, joint source/channel coding, digital communications, and digital signal processing. Professor Fischer has served as Associate Editor for Source Coding for the IEEE Transactions on Information Theory, a
member of the Information Theory Society Board of Governors, Secretary of the Signal Processing and Communication Electronics Technical Committee of the IEEE Communications Society, and program evaluator for ABET. He is a regular reviewer for several journals, and has served on the Program Committee for several Workshops, Symposia, and Conferences. In 1987 Professor Fischer received an outstanding teaching award from the College of Engineering at Texas A&M University. Five times he has received departmental teaching awards at Washington State University. He was a co-recipient of the IEEE Signal Processing Society’s 1993 Senior Award in the Speech Processing Area. In 1996 he was elected Fellow of the IEEE.