LOW COMPLEXITY INDEX-COMPRESSED VECTOR QUANTIZATION FOR IMAGE COMPRESSION

Yu-Chen Hu and Chin-Chen Chang
Department of Computer Science and Information Engineering
National Chung Cheng University, Chiai, Taiwan 621, R.O.C.
E-mail: [martin, ccc]@cs.ccu.edu.tw

Abstract
This paper proposes a novel lossless index compression algorithm that explores the interblock correlation in the index domain and the property of the codebook ordering. The goal of this algorithm is to improve the performance of the VQ scheme at low bit rate while keeping low computation complexity.

In this algorithm, the closest codeword in the codebook is searched for each input vector. Then, the resultant index is compared with the previously encoded indices in a predefined search order to see whether the same index value can be found in the neighboring region. Besides, the relative addressing technique is employed to encode the current index if the same index value cannot be found in the neighboring region. According to the experimental results, the newly proposed algorithm achieves significant reduction of bit rate without introducing extra coding distortion. It is concluded that our algorithm is very efficient and effective for image vector quantization.

1. Introduction
Recently, vector quantization (VQ) [1-4] has been dominantly employed to compress digital images. In general, it can be classified into memoryless VQ and memory VQ. In a memoryless VQ, each block is encoded independently and the index of its corresponding closest codeword in the codebook is transmitted to the decoder. The decoder can then reconstruct the encoded image according to the received indices. Compression is achieved by sending the indices of the closest codewords in the codebook instead of the image vectors themselves.

In the VQ scheme, the closest codeword in the codebook for each input image vector needs to be searched in the encoding procedure. In other words, the closest codeword search process is very time-consuming. To reduce the high computational cost needed in the encoding process, techniques such as the partial distortion search (PDS) [5], the triangle inequality rule (TIE) [6], the mean-distance-sorted partial codebook search (MPS) [7], and the diagonal axes method (DAM) [8] have been developed.

In general, the performance of image quality depends on the codebook size used in the encoding and decoding procedures. Higher performance can be obtained when a larger-sized codebook is used in a VQ scheme. However, a higher bit rate is needed at the same time. One approach of improving the memoryless VQ at low bit rate involves the use of the standard coding schemes such as Huffman coding [9] or arithmetic coding [10]. However, this requires additional computational cost to achieve this goal. Besides, several schemes such as search order coding (SOC) [11], predictive mean search algorithm (PMS) [12], and Index-compressed VQ (ICVQ) [13] exploring the inter-index correlation to improve the memoryless VQ at low bit rate have been developed.

The class of memory VQ's introduces another approach to improve the image quality at low bit rate. A number of memory VQ's that explore the interblock correlation such as finite state VQ (FSVQ) [14-17], address VQ (AVQ) [18]-[20], and predictive VQ [21] have been proposed. The FSVQ encodes each block by searching the closest codeword from the sub-codebook. Therefore, additional coding error may occur and more computation cost is consumed in FSVQ. On the contrary, the AVQ does not introduce extra coding error by building an address codebook containing the indices of the four adjacent blocks. In other words, AVQ reduces the bit rate at the expense of a great amount of computation cost. In fact, most of the memory VQ's are very complicated and require more computation cost than the memoryless VQ.

A low complexity index-compressed VQ is proposed in this paper. The goal of this method is to improve the performance of image quality of memoryless VQ at low bit rate while keeping very low computational complexity. The main idea behind this scheme is that it explores the similarity in the index domain and the property of codebook ordering. Since the neighboring blocks have high correlation, the offset between
two neighboring blocks is thus small. Therefore, the relative addressing technique can be employed to encode the indices to further reduce the bit rate. Simulation results show that the newly proposed algorithm obtains significant reduction of transmitted bits without introducing extra coding error. Besides, this newly proposed algorithm is very simple and suitable for hardware implementation.

The rest of this paper is organized as follows. In Section 2, the mean-distance-ordered partial codebook search algorithm (MPS) will be reviewed. In Section 3, the newly proposed algorithm using mean-sorted codebook will be proposed. In Section 4, experimental results show this newly proposed algorithm is indeed suitable for lossless compression of VQ index. Discussions and conclusions will be given in Section 5.

2. Review on the MPS Algorithm

In 1993, Ra and Kim [7] first proposed the mean-distance-ordered partial codebook search algorithm (MPS). In the MPS method, a new distance measure based on squared Euclidean distance is introduced. Besides, the codebooks used in the MPS method are sorted by the mean values of the codewords. To be more specific, the squared Euclidean distance (SED) between the input image vector \( x \) and the codeword \( y \) in the codebook is redefined as

\[
d_{2}(x, y) = \sum_{j=1}^{k} (x_j - y_j)^2.
\]

Note that \( x_j \) and \( y_j \) denote the \( j \)-th elements of \( x \) and \( y \), respectively, and \( k \) is the dimensionality of each vector.

The new distance measure, i.e. the squared mean distance (SMD), involves the use of the mean value of each block is defined as

\[
d_{m}(x, y) = \left( \sum_{j=1}^{k} x_j \cdot \sum_{j=1}^{k} y_j \right)^2.
\]

One important observation in real images is that the minimal SED codeword is usually in the neighborhood of the minimal SMD codeword. This is because the sum value, i.e. the mean value, of the constituent pixel elements of an arbitrary codeword \( y \) is by itself a sufficiently near statistic for the Euclidean distance measure. In most images, the gray levels of the neighboring pixels are highly concentrated about their mean value. The relationship between SED and SMD measures for any codeword \( y \) is described below

\[
d_{2}(x, y) \leq k \times d_{m}(x, y).
\]

In order to find the closest codeword for image vector \( x \), the SMD value of any codeword \( y \) further in search must be computed. If its SMD value is larger than \( k \) times the minimal SED found so far, i.e. \( d_{2}(x, y_{\text{min}}) \), may safely be rejected.

\[
k \times d_{2}(x, y_{\text{min}}) \leq d_{m}(x, y).
\]

This inequality described in (4) can be used to reject some unnecessary codewords in the search procedure. If the SMD value of codeword \( y \) to be searched is larger than \( k \) times the minimal SED found this far, codeword \( y \) can not be the closest codeword because the inequality \( d_{2}(x, y_{\text{min}}) \leq d_{2}(x, y) \) can be induced from (3) and (4). So, without actually computing the SED value of codeword \( y \), some impossible codewords can be thus rejected.

3. The Proposed Algorithm

In other words, an index map for this image is generated and transmitted to the decoder. Each entry of the index map corresponds to a 4×4 block of pixels. Since the neighboring blocks in one image have high correlation, most of the neighboring blocks are very similar.

To utilize the interblock correlation, a simple and efficient algorithm is proposed. In this algorithm, the codebook is firstly sorted by the mean values of the codewords. Under such arrangement, the neighboring blocks will tend to have the same or approximately the same indices. In other words, the distribution of the indices in the index map is more compact than when the unsorted codebook is used. In this scheme, the indices in the index map can be classified into three different groups. Each group of indices is encoded using its corresponding rule.

Generally, an image is vector-quantized block by block in a raster scan order, that is, from left to right and top to bottom. In other words, the indices in the index map are generated in the raster scan order. In this algorithm, each index in the index map is processed in the raster scan order, too. First, the current processed index is checked to see whether its adjacent left entry and its adjacent upper entry in the index map have the same index value as it. If the same index is found in either of the two positions, only one bit will be used to indicate which one of the two entries has the same index value as the current processed index.

Suppose that \( r \) bits are needed to represent the index, that is, there are \( 2^r \) codewords in the
codebook. In transmission, 2 or \((r + 1)\) bits is required for the current processed index having the same index value or not having the same index, respectively. Note that an extra indicator bit is needed to distinguish whether the same index value of the current index has been found or not.

If both entries do not have the same index value as the current index, the current index is then checked to see whether the relative addressing technique can be employed or not. To accommodate the use of the relative addressing technique, the codebooks used are sorted prior by the mean values of the codewords. By using the mean-sorted codebook, the resultant index map tends to be more compact. In other words, the neighboring indices are quite similar to each other and the differences between any two of them are small.

To make use of the compact distribution of the indices in the index map, the relative addressing technique is employed to improve the compression performance. For each index that does not have the same index value as its adjacent left index and its adjacent upper index, the offset between this index and its previously processed index is computed. If the offset is smaller than the predefined threshold, the relative offset is transmitted to the decoder. Otherwise, the original index for this index is transmitted to the decoder. Note that another extra bit is needed in transmission to indicate the two different types.

The Proposed Algorithm:

**Step 1:** Employ the MPS method to find the closest codeword in the codebook for each input vector \(x\). Let \(I(x)\) denotes the corresponding closest index in the codebook of the current encoding vector.

**Step 2:** If either the adjacent left index or the adjacent upper index of the current processed index \(I(x)\) has the same index value as it, then the index \(I(x)\) is classified into the group 1 and go to Step 5.

**Step 3:** Compute the offset between the previous processed index and \(I(x)\).

**Step 4:** If the offset is smaller than the predefined displacement threshold \(\delta\), then the index \(I(x)\) is classified into the group 2. Otherwise, \(I(x)\) is classified into the group 3.

**Step 5:** Encode the current processed index \(I(x)\) according to the following three rules:

**Rule 1:** If \(I(x)\) belongs to the group 1, then 2 bits are used to represent the two conditions that the same index value is found. The bit stream 00 is transmitted when the adjacent left index of \(I(x)\) has the same index value as \(I(x)\). Otherwise, the bit stream 01 is transmitted.

**Rule 2:** If \(I(x)\) belongs to the group 2, the bit stream \(conc(10,y)\) is transmitted to the decoder. Here, the variable \(y\) denotes the binary representation of the relative offset computed in Step 3, and \(conc(u,v)\) means the bit stream that is obtained by concatenating the tow bit streams \(u\) and \(v\).

**Rule 3:** If \(I(x)\) belongs to the group 3, the bit stream \(conc(11,z)\) is transmitted. Here, the variable \(z\) denotes the binary representation of the index value of \(I(x)\).

In the decoder, the image blocks are constructed in a raster scan order. The first two bits of the received index are used to justify to which group the current decoded index belongs. More precisely, the first received bit can be used to justify whether the same index value of the current decoded index has been found in its adjacent left index and its adjacent upper index. According to the first two received bits, the current index can be easily reconstructed and thus the encoded block can be reconstructed by executing a simple table look-up operation. The decoding algorithm is very simple and thus omitted here.

4. Experimental Results

To verify the performance of this newly proposed algorithm, a variety of simulations have been performed. All our experiments are performed on SUN Sparc10 Workstation. Five monochromatic images of 512×512 pixels with 256 gray levels are used to design the codebooks using the LBG algorithm [1]. The block size is 4×4 and the codebook size is \(N_c\). In other words, the length of the VQ index in the traditional VQ scheme equals \(r\), where \(r=\log_2 N_c\). Therefore, the bit rate of the traditional VQ system equals \(r/k\), where \(k\) is the vector dimension. The bit rate of the proposed algorithms is given by

\[
BR=[Tno(g_j) \times (1+1)+Tno(g_j) \times (2+s) + Tno(g_j) \times (2+r)]/(M \times N),
\]

where

\[
Tno(g_j) = \text{the total number of indices that belong to the group } j, \text{ where } 1 \leq j \leq 3,
\]

\(s\) = the number of bits assigned to relative addressing coding, and

\(M \times N\) = the image size.
Table 1
Comparative performance of bit rate using a VQ codebook with 128 codewords

<table>
<thead>
<tr>
<th>Images</th>
<th>Lenna</th>
<th>F-16</th>
<th>Toys</th>
<th>Tiffany</th>
<th>Averaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPCM/Huf</td>
<td>0.3656</td>
<td>0.3209</td>
<td>0.2251</td>
<td>0.2427</td>
<td>0.2885</td>
</tr>
<tr>
<td>MPS [18]</td>
<td>0.3761</td>
<td>0.3534</td>
<td>0.3246</td>
<td>0.3330</td>
<td>0.3467</td>
</tr>
<tr>
<td>SOC [9]</td>
<td>0.2806</td>
<td>0.2804</td>
<td>0.2316</td>
<td>0.2264</td>
<td>0.2547</td>
</tr>
<tr>
<td>Method 1</td>
<td>0.2651</td>
<td>0.2443</td>
<td>0.1806</td>
<td>0.1786</td>
<td>0.2171</td>
</tr>
<tr>
<td>Method 2</td>
<td>0.2583</td>
<td>0.2438</td>
<td>0.1795</td>
<td>0.1775</td>
<td>0.2146</td>
</tr>
</tbody>
</table>

Table 2
Comparative performance of bit rate using a VQ codebook with 256 codewords

<table>
<thead>
<tr>
<th>Images</th>
<th>Lenna</th>
<th>F-16</th>
<th>Toys</th>
<th>Tiffany</th>
<th>Averaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPCM/Huf</td>
<td>0.4161</td>
<td>0.3725</td>
<td>0.2509</td>
<td>0.2774</td>
<td>0.3167</td>
</tr>
<tr>
<td>MPS [18]</td>
<td>0.4347</td>
<td>0.4196</td>
<td>0.3845</td>
<td>0.3979</td>
<td>0.4091</td>
</tr>
<tr>
<td>SOC [9]</td>
<td>0.3368</td>
<td>0.3211</td>
<td>0.2545</td>
<td>0.2518</td>
<td>0.2910</td>
</tr>
<tr>
<td>Method 1</td>
<td>0.3233</td>
<td>0.2869</td>
<td>0.2050</td>
<td>0.2070</td>
<td>0.2555</td>
</tr>
<tr>
<td>Method 2</td>
<td>0.3149</td>
<td>0.2864</td>
<td>0.2041</td>
<td>0.2056</td>
<td>0.2527</td>
</tr>
</tbody>
</table>

Table 3
Comparative performance of bit rate using a VQ codebook with 512 codewords

<table>
<thead>
<tr>
<th>Images</th>
<th>Lenna</th>
<th>F-16</th>
<th>Toys</th>
<th>Tiffany</th>
<th>Averaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPCM/Huf</td>
<td>0.4856</td>
<td>0.4623</td>
<td>0.3979</td>
<td>0.3752</td>
<td>0.4259</td>
</tr>
<tr>
<td>MPS Method</td>
<td>0.4895</td>
<td>0.4511</td>
<td>0.4308</td>
<td>0.4186</td>
<td>0.4475</td>
</tr>
<tr>
<td>SOC Method</td>
<td>0.4209</td>
<td>0.4126</td>
<td>0.3207</td>
<td>0.3842</td>
<td>0.3846</td>
</tr>
<tr>
<td>Method 1</td>
<td>0.4060</td>
<td>0.3803</td>
<td>0.2788</td>
<td>0.3498</td>
<td>0.3357</td>
</tr>
<tr>
<td>Method 2</td>
<td>0.3963</td>
<td>0.3823</td>
<td>0.2787</td>
<td>0.3464</td>
<td>0.3509</td>
</tr>
</tbody>
</table>

Tables 1, 2, and 3 show the results of the proposed algorithm and three other methods using a mean-ordered VQ codebook sized 128, 256, and 512, respectively. The DPCM/Huf method employs the DPCM method to remove the correlation among indices and follow the Huffman Coding technique to encode the residual magnitudes. The PMS method [12] uses the relative addressing technique to send the indices to the decoder using mean-order codebook. The SOC method [11] introduces the search-order coding technique to compress the VQ index. Besides, our method 1 and method 2 use the adjacent left index and the adjacent upper index of the current processed index to compute the relative offset, respectively.

In our method 2, the averaged bit rates are 0.2171, 0.2527, and 0.3509 bit/pixel for the codebooks sized 128, 256, and 512, respectively. Thus, the bit rate reduction is 50.38% for the codebook sized 128. Furthermore, the bit rate reductions for codebooks sized 256 and 512 are 49.46%, and 37.62%, respectively. It is interesting that the bit rate reduction of the codebook sized 128 is greater than that of the codebooks sized 256 and 512, respectively. This may be due to the higher interblock correlation in the resultant index map owing to the smaller codebook. Besides, according to the experimental results, our method 2 has slightly better performance than that of our method 1.

Table 4 shows the coding results of the percentage of the indices that either its adjacent left index or its adjacent upper index has the same index value as it. The image qualities of these five 512×512 images are listed in Table 5. It is interesting that the image qualities using 256 codewords are better than those 128 codewords are used. The same scenario holds when 512 codewords are used. Furthermore, the total number of the same index value found in the codebook sized 128 is larger than those found in codebooks sized 256 and 512, respectively. This is due to the higher inter-index correlation in the codebook sized 128.

Table 4
Results of the percentage of the indices having the same index value

<table>
<thead>
<tr>
<th>Sizes</th>
<th>128 codewords</th>
<th>256 codewords</th>
<th>512 codewords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images</td>
<td>Lenna</td>
<td>F-16</td>
<td>Toys</td>
</tr>
<tr>
<td>128</td>
<td>60.66%</td>
<td>99.02%</td>
<td>36.99%</td>
</tr>
<tr>
<td>256</td>
<td>66.30%</td>
<td>59.46%</td>
<td>41.71%</td>
</tr>
<tr>
<td>512</td>
<td>83.28%</td>
<td>79.77%</td>
<td>63.61%</td>
</tr>
<tr>
<td>Averaged</td>
<td>72.94%</td>
<td>66.07%</td>
<td>46.76%</td>
</tr>
</tbody>
</table>

Table 5
Experimental result of the image qualities using different sizes of codebooks

<table>
<thead>
<tr>
<th>Sizes</th>
<th>128 codewords</th>
<th>256 codewords</th>
<th>512 codewords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images</td>
<td>Lenna</td>
<td>F-16</td>
<td>Toys</td>
</tr>
<tr>
<td>128</td>
<td>28.901 dB</td>
<td>30.639 dB</td>
<td>31.840 dB</td>
</tr>
<tr>
<td>256</td>
<td>28.339 dB</td>
<td>29.327 dB</td>
<td>30.745 dB</td>
</tr>
<tr>
<td>512</td>
<td>27.674 dB</td>
<td>28.268 dB</td>
<td>31.552 dB</td>
</tr>
<tr>
<td>Averaged</td>
<td>28.191 dB</td>
<td>29.435 dB</td>
<td>31.229 dB</td>
</tr>
</tbody>
</table>

Table 6
Execution times (in seconds) of the proposed algorithm and various methods using different codebook sizes

<table>
<thead>
<tr>
<th>Sizes</th>
<th>128 codewords</th>
<th>256 codewords</th>
<th>512 codewords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images</td>
<td>Lenna</td>
<td>F-16</td>
<td>Toys</td>
</tr>
<tr>
<td>FS-VQ</td>
<td>23.683</td>
<td>47.381</td>
<td>95.333</td>
</tr>
<tr>
<td>MPS [12]</td>
<td>1.820</td>
<td>2.859</td>
<td>3.900</td>
</tr>
<tr>
<td>Method 1</td>
<td>1.773</td>
<td>2.823</td>
<td>3.883</td>
</tr>
<tr>
<td>Method 2</td>
<td>1.803</td>
<td>2.883</td>
<td>3.933</td>
</tr>
</tbody>
</table>

Table 6 shows the results of execution time using different codebook sizes. Note that FS-VQ denotes the memoryless VQ scheme with full search algorithm. According to the results shown in Table 6, it is obvious that each of the
two new methods requires very little cost of computation, compared to FS-VQ. Besides, the newly proposed algorithm is very simple and suitable for hardware implementation.

To achieve higher compression ratio, the variable length coding (VLC) technique can be applied to encode the output messages of our newly proposed algorithm. However, more computation complexity is required to implement this VLC operation. According to the experimental results, our algorithm achieves significant saving of bit rate without introducing extra coding error. Furthermore, it is very simple and suitable for hardware implementation.

4. Conclusions

A low complexity index-compressed vector quantization scheme is proposed in this paper. The goal of this scheme is to improve the performance of the VQ scheme at low bit rate. Since the class of memory VQ's generally requires a large amount of computational cost. This scheme adopts the structure of the memoryless VQ and compresses the resultant indices.

According to the experimental results, it is shown that the reduction of bit rate is 50.4% codebook sized 128, compared to the traditional memoryless VQ scheme. Besides, the reductions of bit rates are 49.5% and 37.6% for the codebooks sized 256 and 512, respectively. In other words, significant reductions of bit rate without introducing extra coding error by using the newly proposed methods. Further, this algorithm requires very little computational complexity. Therefore, these it is very efficient and suitable for hardware implementation.

References


