AN IMPROVED FAST ENCODING METHOD FOR VECTOR QUANTIZATION
BASED ON MEMORY-EFFICIENT DATA STRUCTURE

Zhibin Pan¹, Koji Kotani², and Tadahiro Ohmi¹

1) New Industry Creation Hatchery Center, Tohoku University, Japan
2) Department of Electronic Engineering, Graduate School of Engineering, Tohoku University, Japan

Aza-aoba 10, Aramaki, Aoba-ku, Sendai, 980-8579, Japan
E-mail: pzb@iff.niche.tohoku.ac.jp

ABSTRACT

In the framework of vector quantization (VQ), fast search method is a key issue because it is the time bottleneck in VQ encoding process. To speed up VQ, some very effective fast search methods that are based on using statistical features (i.e. the mean, the variance and L2 norm) of a k-dimensional vector have already been proposed in previous works [2], [3], [4]. It is rather easy to obtain the mean of an input vector on-line. However, in order to obtain the variance of an input vector have already been proposed in previous works [2], [3], [4]. It is rather easy to obtain the mean of an input vector on-line. However, in order to obtain the variance of an input vector, it needs (2k−1) additions (±), k multiplications (×) and once square root (sqrt) operation on-line. Similarly, to obtain L2 norm of an input vector, it also needs (k−1) additions (±), k multiplications (×) and once square root (sqrt) operation on-line. Clearly, these operations are a rather heavy overhead in the search process. In addition, all computations in a search process must use real rather than integer value. For each codeword, three extra memories are necessary to store its features.

To solve the overhead problem of computing the variance and L2 norm of an input vector on-line and to make all computations in integer form possible, the previous works [5], [6] proposed to use the sum and two partial sums only as features of a vector for reducing search space. But they need three extra memories as well for each codeword to store the features. In order to avoid this extra memory requirement, this paper proposes a memory-efficient data structure for storing the sum, two partial sums and the original vector to further improve the previous works [5], [6]. Meanwhile, a more efficient computation method for rejection tests is also developed. In addition, all computations can be realized in integer form. Experimental results confirmed the proposed method outperforms the previous works obviously.

1. INTRODUCTION

Vector quantization (VQ) is a widely used asymmetric signal compression method [1]. In conventional VQ method, an N×N image to be encoded is firstly divided into a series of non-overlapping smaller n×n image blocks. Then VQ encoding is implemented block by block sequentially. The distortion between an input image block and a codeword can be measured by squared Euclidean distance for simplicity as

\[ d^2(I, C_i) = \sum_{j=1}^{n} (I_j - C_{i,j})^2 \quad i = 1, 2, \cdots, N_c \]  

where I is the current image block, \( C_i \) is the ith codeword, \( j \) represents the jth element of a vector, \( k (=n^2) \) is the vector dimension and \( N_c \) is the codebook size.

Then a best-matched codeword with minimum distortion, which is called winner afterwards, can be determined straightforwardly by

\[ d^2(I, C_{w}) = \min_{i} [d^2(I, C_i)] \quad i = 1, 2, \cdots, N_c \]  

where \( C_w \) means winner. And subscript “w” is the index of the winner. This process for finding winner is called full search (FS) because the matching is executed over the whole codebook. FS is very heavy due to it performing \( N_c \) times k-dimensional Euclidean distance computation. Once “w” has been found, which conventionally uses much less bits than an original codeword, VQ only transmits this index “w” instead of the codeword \( C_w \) to reduce the amount of image data in order to realize image compression. Because the same codebook has also been stored at the receiver, by using the received index “w”, it is very easy to reconstruct an image by pasting the corresponding codeword one by one.

Because the high dimension k of a vector is the main problem for fast VQ encoding, it is very important to use lower dimensional features to approximately express a vector in order to reject obviously unlikely codewords by just a little cost. There exist many fast search methods for VQ already. One class of them [2]-[4] is based on using scalar statistical features of a vector (the mean or L1 norm, the variance and L2 norm), which can roughly describe a vector by just 1 dimension, to compute the difference between two vectors in order to reject a candidate codeword to avoid computing Euclidean distance. Another class of them [5], [6] is based on the multi-resolution concept by dividing a k-dimensional vector in half to construct a pair of more precise features over two partial vectors to roughly describe the original vector in order to make a rejection. Both classes of methods are very search-efficient but they suffer from extra memory requirements and the overhead problem of computation once a rejection test fails. This paper aims at improving the method using the sum and two partial sums proposed in [5], [6] by completely omitting extra memory requirement and reusing previously computed difference values to overcome the overhead of computation.

2. RELATED PREVIOUS WORKS

During a winner search process, suppose the found “so far” minimum Euclidean distance is \( d_{\text{min}} \). Based on statistical
features of a vector, the previous work [2] proposed a
codeword rejection rule as: If \( k(MI - MC) - d_{min} \) holds,
then reject \( C_i \) safely, where MI is the mean of an input image
vector I and MC_i means the same for \( C_i \). This is the famous
ENNS method. Then, the previous work [3] proposed a
supplementary codeword rejection rule when ENNS method
fails as: If \( k(MI - MC) + (VI - VC) - d_{min} \) holds, then reject
\( C_i \) safely as well, where VI is the variance of I and VC_i means
the same for \( C_i \). This is the famous ENENS method.
To improve [3] further, the previous work [4] proposed another
supplementary codeword rejection rule when ENENS method also
fails as: If \( (L_i - L_j, C_i) - d_{min} \) holds, then reject \( C_i \) safely as well, where \( L_i \) is the L2 norm of I
and \( L_jC_i \) means the same for \( C_i \). This is state-of-the-art
ENENS (i.e. equal-average equal-variance equal-norm nearest
neighbor search) method. EEENNS method needs three extra memories for each codeword. Obviously, if any
one of these three rejection tests fails, the obtained difference
value at each step will become an overhead completely and
cannot be reused at all. In addition, it needs \( 2(k-1) \) additions
\( \pm \) k multiplications \( (\times) \) and once square root \( (\sqrt{\cdot}) \)
operation for computing each VI on-line. Similarly, it also
needs \( k-1 \) additions \( \pm \) k multiplications \( (\times) \) and once
square root \( (\sqrt{\cdot}) \) operation for computing each L2 norm
on-line. Mathematically, EEENNS method used both first
order moment, second order central moment (the variance)
and second order raw moment (L2 norm) of a vector as its
features. Clearly, the two second order moments are much
heavier than first order moment for on-line computation.
On the other hand, the previous works [5], [6] proposed to
use the sum (first resolution, lower) and partial sums (second
resolution, higher) only as appropriate features to realize a
2-resolution description for a k-dimensional vector as below

\[
S_I = \sum_{j=1}^{k} I_j, \quad S_{I_1} = \sum_{j=1}^{k/2} I_j, \quad S_{I_2} = \sum_{j=2^{k/2+1}}^{k} I_j,
\]

\[
SC_i = \sum_{j=1}^{k} C_{1,i,j}, \quad SC_{i,1} = \sum_{j=1}^{k/2} C_{1,i,j}, \quad SC_{i,2} = \sum_{j=2^{k/2+1}}^{k} C_{1,i,j}
\]

(3)

Based on Eq.3, three differences between I and \( C_i \) as the
estimations of Euclidean distance can be defined as

\[
d_{1,i} = |SI - SC_{i,1}|
\]

\[
d_{2,i} = |SI - SC_{i,1} + S_{I_2} - SC_{i,2}|
\]

\[
d_{22,i} = \sqrt{(SI - SC_{i,1})^2 + (S_{I_2} - SC_{i,2})^2}
\]

(4)

Then, an inequality for rejection tests is proven in [6] as

\[
\sqrt{k} \cdot d(I, C_i)_{test4} \geq \sqrt{2} \cdot d_{22,i} \geq d_{21,i} \geq d_{1,i} \geq \sqrt{k} \cdot d_{min}
\]

(5-1)

\[
kd^2(I, C_i)_{test4} \geq 2d_{22,i}^2 \geq d_{21,i}^2 \geq d_{1,i}^2 \geq kd_{min}^2
\]

(5-2)

\( i = 1, 2, \ldots, N_c \)

Eq.5-1 and Eq.5-2 are equivalent but Eq.5-2 does not need
to use any sqrt operation. Furthermore, all computations for
rejection tests in Eq.5-2 can be conducted only in integer
form, which is beneficial to a higher computation speed and
an easier hardware implementation. Eq.5-2 is the key in
this paper. For a candidate \( C_i \), the test sequence for realizing a
promising rejection is from right to left. In Eq.5-2, the first
rejection test (Test 1) in fact has the same rejection power as
ENNS method that uses the mean of a vector because both the
sum and the mean are \( L_1 \) norm. The last rejection test (Test 4)
is actually a Euclidean distance computation.
Because the features of all codewords \( C_i \) can be computed
and stored off-line, they do not affect the search efficiency
for VQ encoding. To obtain SI, SI_1 and SI_2 for input I on-line,
SI_1 and SI_2 should be computed first. It needs \( 2(k/2-1) \)
\( =k-2 \) additions \( (\pm) \). Then, just once addition \( \pm \) is necessary
obtain SI. As a result, \( (k-1) \) additions \( (\pm) \) in total is
sufficient. In contrast, it needs \( (k-1) \) additions \( (\pm) \) and once
division \( (\div) \) to obtain the mean \( (MI) \) only in ENENS method.
Mathematically, Eq.5-2 only uses first order moment but second order moment of a vector and two
partial vectors, it is rather light for on-line computation.
Because any codeword rejection method is for reducing
the search space, how small it can be reduced is a key issue.
In order to demonstrate how well the search space can be
reduced by using each test in Eq.5-1, we can use the 2 partial
sums to generate a new 2-D coordinate \( u - o - v \) as shown
in Fig.1. The horizontal axis represents the first partial sum
and the vertical axis represents the second partial sum. Then,
each codeword in \( R^k \) space can be mapped into a point in
this 2-D coordinate. Because image data are 8-bit encoded
and partial sums are constructed by equally dividing a vector
into 2 parts, any input image vector I or codeword will be
mapped into a point located in a range with the bounds of
[0~ (k/2)×255, 0~ (k/2)×255] as shown in Fig.1. For a given
input I, it can be plotted as the point “I” in the 2-D partial
sum coordinate as shown in Fig.1. For convenience, Eq.5-1
is used to interpret the effectiveness of each rejection test.
Because winner search in VQ is to find the nearest
codeword to I, we can shift the origin “O” to “I” to have
another new 2-D coordinate \( u' - I - v' \) , which let the point
“O’” be \( (0, 0) \) for simplicity. The boundary of search space
determined by \( d_{1,i} \) test is actually 2 straight lines according to
Eq.5-1. Therefore, we can use 2 points to determine a
straight line. Obviously, the 2 points \((\sqrt{k} \cdot d_{min}, 0)\) and \((0, \sqrt{k} \cdot d_{min})\)
can determine a straight and the 2 points \((- \sqrt{k} \cdot d_{min}, 0)\) and
and \((0, -\sqrt{k} \cdot d_{min})\) can determine another straight line. These
2 straight lines combining with the bounds of [0~ (k/2)×255, 0~ (k/2)×255]
will constitute the boundary of the search space (Trapezoid) reduced by \( d_{1,i} \). Similarly, the reduced
search space by \( d_{21,i} \) test can be plotted as a square and the
reduced search space by \( d_{22,i} \) test can be plotted as a circle
step by step as shown in Fig.1. How small the search space
can be reduced strongly depends on \( d_{min} \) and the dimension k.
This is a geometric explanation on how each rejection test
works in Eq.5-1 (the same for Eq.5-2). It is apparent that the
search space can be reduced greatly by \( d_{21,i} \) and \( d_{22,i} \) check
further. The final search space that needs Euclidean distance
computations is within the circle centering at I with a radius of
\( \sqrt{k/2} \times d_{min} \).
From Fig.1, it is clear that search space can be reduced greatly after the first three rejection tests in Eq.5-1. If a rejection fails ultimately, Euclidean distance must be computed. Then either a rejection judgment is realized or \( d_{\text{min}} \) is updated when a better-matched codeword is found during the search process.

### 3. IMPROVED METHOD

The **first improvement** to the previous work [6] is to avoid using any extra memories for storing all features of a vector together with the original vector. For any \( C_v \), its sum \( SC_i \), two partial sums \( SC_{i,1} \) and \( SC_{i,2} \) are computed first according to the definition in Eq.3 off-line. Then, only the values at solid locations are stored but the values at blank locations are NOT stored (i.e. \( SC_{i,2} \), \( Ci_{k/2} \) and \( Ci_k \)) as demonstrated in Fig.2. Obviously, \( k \) memories in total are sufficient for storing a \( k \)-dimensional \( C_i \) and its three according features this way. There is no memory redundancy in this case because for \( k \) independent values of a vector, only \( k \) memories are used. This is guaranteed in principle by information theory. For input \( I \), it is the same.

![Diagram](image)

From Fig.2, \( SC_{i,2} \), \( Ci_{k/2} \), \( Ci_k \) and \( SL_2 \), \( I_{k/2} \) and \( I_k \) have to be computed on-line when it is necessary. For example, a straightforward method is to compute \( SL_2=(SI_1-SI_2) \) and \( SC_{i,2}=(SC_1-SC_{i,1}) \) first in order to compute \( (SI_2-SC_{i,2}) \) on-line, which needs 3 additions (\( \pm \)) in total. In contrast, a conventional way for computing \( (SI_2-SC_{i,2}) \) just needs once addition (\( \pm \)). It will certainly introduce some overhead of on-line computation. Therefore, the **second improvement** to [6] is to try to avoid this overhead coming from using a data structure shown in Fig.2. A detailed analysis is given below.

In Eq.5-2, Test 1 has not been affected because \( SI \) and \( SC_i \) are stored as usual. If Test 1 failed, Test 2 and Test 3 must be conducted. Because \( (SI_1-SC_i) \) is already known at this moment and \( (SI_1-SC_{i,1}) \) is computed first, Eq.4 can be rewritten as

\[
d_{21,i} = |SI_i - SC_{i,1}| + |SI_i - SC_i - (SI_1 - SC_{i,1})|
\]

\[
d_{22,i} = \sqrt{(SI_i - SC_{i,1})^2 + (SI_i - SC_i - (SI_1 - SC_{i,1}))^2}
\]

Obviously, the term \((SI_1-SC_{i,1})\) also requires just once addition (\( \pm \)) so that Eq.6 needs exact the same computational cost as Eq.4. It benefits from reducing the values of \( (SI_1-SC_1) \) and \( (SI_1-SC_{i,1}) \). As a result, no additional on-line computational cost occurs due to introducing the memory-efficient data structure to Test 1~Test 3.

If all of Test 1~Test 3 failed, Test 4 must be conducted by computing Euclidean distance. A straightforward method needs to compute \( Ci_{k/2}, Ci_k \) first, which needs \( 2\times(k-2) \) on-line additions (\( \pm \)) in total. This is an unavoidable on-line additional computational cost. However, based on the concept of partial distortion search (PDS) [7], it is profitable to test the distortion by using the stored \([1-\ldots-(k/2-1)]\) and \([1-\ldots-(k/2-1)]\) elements first as defined in Eq.7

\[
T^2(i, Ci_i) = \sum_{j=1}^{k/2} (I_{1,j} - Ci_{1,j})^2 + \sum_{j=1}^{k/2} (I_{2,j} - Ci_{2,j})^2 \geq \min_{T_{\text{on-line}}} \text{d}_{\text{min}}^2
\]

Furthermore, for any \( m \)-dimensional vector \( X, \sqrt{\sum_{i=1}^{m} |X_i|^2} \geq |X| \) always holds (Cf. Eq.7 in [6]), then the **third improvement** to [6] is to integrate another new rejection test as

\[
\tilde{d}^2(i, Ci_i) = \left[ \sum_{j=1}^{k/2} (I_{1,j} - Ci_{1,j})^2 + \sum_{j=1}^{k/2} (I_{2,j} - Ci_{2,j})^2 \right] \geq \min_{T_{\text{on-line}}} \text{d}_{\text{min}}^2
\]

In practice, Test 3-2 should be carried out before Test 3-1 because it does not need any heavier multiplications (\( \times \)). Besides, for the later Test 3-1, it does not need to compute \( (I_i-Ci) \) for all \( j \) again. If Test 3-2 or Test 3-1 is successful, the candidate \( Ci_i \) can be rejected safely so that no additional on-line computational cost occurs due to introducing the memory-efficient data structure to Test 1~Test 3, Test 3-1 and Test 3-2 till now. This benefits from a suitable order for computing distortions.

If Test 3-1 also fails ultimately, Test 4 must be conducted but by Eq.9 instead of Eq.1 as

\[
d^2(i, Ci_i) = \tilde{d}^2(i, Ci_i) + \left[ \sum_{j=1}^{k/2} (I_{1,j} - Ci_{1,j})^2 + \sum_{j=1}^{k/2} (I_{2,j} - Ci_{2,j})^2 \right] \geq d_{\text{min}}^2
\]

where \( Ci_{1,j} = SC_{i,1} - \sum_{l=1}^{j-1} Ci_{l,j} \) and \( Ci_{2,j} = SC_{i,2} - SC_{i,1} - \sum_{l=1}^{j-1} Ci_{l,j} \), which needs \( (k-2) \) on-line additions (\( \pm \)) for each \( Ci_i \). For
input I, it is similar but only needs to compute I_2 and I_1 once. Because T(I, C) is already known, Eq.9 is much lighter than Eq.1. If Eq.9 holds, reject C_i the same; Otherwise, update d_{min}^2 by current d^2(I, C_i) and the according index of “so far” best-matched codeword.

4. EXPERIMENTAL RESULTS

Simulation experiments with MATLAB are conducted on a personal computer. Block size is 4×4. Three codebooks are generated by using a modified Kohonen’s self-organizing map (SOM) method [8] with the 512×512, 8-bit Lena image as a training set to test search efficiency. Each codebook is sorted along the sums of codewords in an ascending order and SC_i is computed and stored off-line. Winner search starts from an initial best-matched codeword C_p, which is closed to the input I in terms of a minimum sum difference d_{lp}. C_p can be determined by a binary search process, which needs log_2(N_c) times comparisons (Cmp).

Based on the analysis in Section 3, it is clear that the proposed method is more efficient than the method used in [6] because a new rejection test (Eq.8) is integrated. On the other hand, if a vector is viewed as a set with k samples, it is very natural to use the statistical features to describe this sample set and then to distinguish two similar sample sets by them. That is why EEENNS method is widely used in VQ encoding field. In this paper, EEENNS method combined with a new rejection test \( k \sum |x_j - c_{j1}|^2 \geq \sum |x_j - c_{j1}|^2 \geq kd_{min}^2 \) similar to Eq.8 is used as a benchmark for a performance comparison, which is evaluated in necessary arithmetical operations per input vector with FS as a relative baseline.

From Table 1, it is clear that the proposed method is more search-efficient and memory-efficient as well than EEENNS method. It is a promising method for fast VQ encoding.

5. CONCLUSIONS

In this paper, an improved fast VQ search method based on the sum and two partial sums of a vector is proposed. Three advantages are realized. First, a memory-efficient data structure is introduced that makes k memories be sufficient for storing all information of a k-dimensional vector. Second, overhead is avoided by reusing the obtained previous difference values during a search process. Third, a new rejection test is integrated as well before computing Euclidean distance. In addition, the proposed method does not use any square root (Sqrt) operations and makes all Euclidean distance computations in integer form possible. The proposed method outperforms state-of-the-art EEENNS method obviously.

6. REFERENCES