Fast Search Method for Image Vector Quantization Based on Equal-Average Equal-Variance and Partial Sum Concept

Zhibin Pan[1), Koji Kotani[2), and Tadahiro Ohmi[1)

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@fff.niche.tohoku.ac.jp

ABSTRACT

The encoding process of image vector quantization (VQ) is very heavy due to it performing a lot of k-dimensional Euclidean distance computations. In order to speed up VQ encoding, it is most important to avoid unnecessary exact Euclidean distance computations as many as possible by using features of a vector to estimate how large it is first so as to reject most of unlikely codewords. The mean, the variance, L_2 norm and partial sum of a vector have been proposed as effective features in previous works for fast VQ encoding.

Recently, in the previous work [6], three features of the mean, the variance and L_2 norm are used together to derive an EEENNS search method, which is very search efficient but still has obvious computational redundancy. This paper aims at modifying the results of EEENNS method further by introducing another feature of partial sum to replace L_2 norm feature so as to reduce more search space. Mathematical analysis and experimental results confirmed that the proposed method is more search efficient compared to [6].

1. INTRODUCTION

Vector quantization (VQ) [1] is a classical method for image compression. Conventionally, VQ uses a look-up table (called codebook) principle by template matching so as to find the closest item (winner) to an input image block in the whole table according to a certain distortion measurement (usually Euclidean distance for simplicity). Then, VQ only transmits winner index instead of winner itself to reduce the amount of image data because a codeword index uses much less bits than a codeword. Since the exact same table (codebook) is pre-stored at the receiver, an image can be decoded easily via sequentially pasting codewords that are retrieved at the receiver by using the received winner indexes. Therefore, VQ has a very heavy encoding process due to a lot of k-dimensional (k-D) Euclidean distance computations for matching and a very simple decoding process. For practical applications, one of the primary limitations to VQ encoding is its computational complexity.

There have been many fast search algorithms developed for VQ encoding already. Concerning about using lower dimensional feature to make an estimation so as to avoid exact Euclidean distance computation for obviously unlikely codewords, the most straightforward and most famous features are the statistical features of a vector (c.f. a k-dimensional vector can also be viewed as a k-element sample set), which are the mean or the sum [2], the variance [3] and L_2 norm [4]. Based on these three conventional statistical features, several fast VQ encoding methods [5], [6] have been proposed by using a combination among them. However, there are still computational redundancies in them.

This paper proposes to use partial sum [7], [8], [9] as an additional feature to enhance the results in previous works [2]-[6] and then uses state-of-the art EEENNS method in [6] as a benchmark for performance comparison. Experimental results confirmed that the computational cost of the proposed method can be reduced to a great extent for typical standard images compared to the EEENNS method in the previous work [6] with full search (FS) as a relative baseline.

Conventionally, an N×N image to be encoded by VQ method is firstly divided into a series of non-overlapping smaller n×n image blocks. Then VQ encoding is executed block by block sequentially.

For an image block encoded by VQ, the real distortion is a difference vector as defined below

$$D_i = x - y_i = \left[D_{i,1}, D_{i,1}, \cdots, D_{i,k}\right]^T \qquad i = 1,2,\cdots,N_c \qquad (1)$$

where $x=[x_1, x_2, \ldots, x_k]^T$ is the current input image block, $y_i=[y_{i,1}, y_{i,2}, \ldots, y_{i,k}]^T$ is the i^{th} codeword in the codebook $Y=\{y_i \mid i=1,2, \ldots, N_c\}$, the superscript "^{T}" is a transpose operation, k (k=n×n) is the vector dimension and N_c is the codebook size.

For simplicity, the real distortion D_i is usually measured by squared Euclidean distance as

$$d^2(x, y_i) = \left(\|D_i\|_2\right)^2 = \sum_{j=1}^{k}\left(x_j - y_{i,j}\right)^2 \qquad i = 1,2,\cdots,N_c \qquad (2)$$

where d(. , .) is a function for computing Euclidean distance, $\| \cdot \|_2$ means L_2 norm of a vector and j represents the j^{th} dimension of a vector.

When Euclidean distance is used as the final distortion measurement, the winner can be determined directly by

$$d^2(x, y_w) = \min_{y_i \in Y}\left[d^2(x, y_i)\right] \qquad i = 1,2,\cdots,N_c \qquad (3)$$

where the best-matched codeword y_w is called as the winner and its subscript "w" is the winner index. Then the index "w" instead of y_w is transmitted to the decoder for realizing data compression. This is full search (FS) over the codebook $Y=\{y_i \mid i=1,2, \ldots, N_c\}$. FS method can achieve the best PSNR for a fixed codebook but it is computationally very heavy due to N_c times Euclidean distance computations.

2. RELATED PREVIOUS WORK

For a k-dimensional image vector, its most popular statistical features are the mean or the sum (equivalent to its L_1 norm), the variance and L_2 norm. In other words, a vector can be approximately described by its features from the statistics point of view. Therefore, in order to measure how different 2 vectors are with just a little computational cost instead of an

0-7803-9332-5/05/$20.00©2005 IEEE

immediate heavy Euclidean distance computation, the difference between the corresponding features can be used first to estimate Euclidean distance. Because VQ encoding only needs to find out the minimum Euclidean distance so as to determine a sole winner y_w rather than all Euclidean distances from the input x to all codewords y_i, it allows to introduce an estimation for Euclidean distance, which must be less or equal to Euclidean distance. Therefore, if this estimation is sufficiently large, it can guarantee that the corresponding Euclidean distance will surely be larger than the current minimum Euclidean distance as well. Then, a rejection to the current codeword y_i is permitted so that the exact Euclidean distance computation can be completely avoided to reduce computational burden.

Suppose the "so far" minimum Euclidean distance is d_{min}, the previous work [2] proposed a codeword rejection rule by using the mean information as

$$k(Mx - My_i)^2 \geq d_{\min}^2 \tag{4}$$

where $Mx = \sum_{j=1}^{k} x_j / k$ is the mean of an input image block x and $My_i = \sum_{j=1}^{k} y_{i,j} / k$ means the same for y_i. If Eq.4 holds, then reject y_i safely. This is the famous ENNS (i.e. equal-average nearest neighbor search) method. Eq.4 needs one extra memory to store My_i for each y_i and one "±", one "×" (Eq.4 practically tests $(Mx - My_i)^2 \geq d_{\min}^2 / k$ instead when $d_{\min}^2 / k$ is stored as a whole) and one "Cmp" (comparison) operation for a rejection test.

To improve [2], the previous work [3] further proposed a codeword rejection rule by using the variance information as

$$(V_x - V_{y_i})^2 \geq d_{\min}^2 \tag{5}$$

where $V_x \overset{Def}{=} \sqrt{\sum_{j=1}^{k} (x_j - Mx)^2}$ is the variance of input x and $V_{y_i} \overset{Def}{=} \sqrt{\sum_{j=1}^{k} (y_{i,j} - My_i)^2}$ means the same for y_i. If Eq.5 holds, then reject y_i safely. This is the famous EENNS (i.e. equal-average equal-variance NNS) method. Eq.5 needs one extra memory to store V_{yi} for each y_i and one "±", one "×" and one "Cmp" operation for a rejection test.

To improve [3], the previous work [5] proposed another codeword rejection rule by using both the mean and the variance information simultaneously as

$$k(Mx - My_i)^2 + (V_x - V_{y_i})^2 \geq d_{\min}^2 \tag{6}$$

If Eq.6 holds, then reject y_i safely. Eq.6 is more powerful than Eq.4 or Eq.5. In practice, Eq.6 tests $k(Mx - My_i)^2 \geq d_{\min}^2$ by using Eq.4 first in order to avoid a possible overhead of computation, which means to avoid computing $(V_x - V_{y_i})^2$ because a lot of codewords can be easily rejected by this first test. This is IEENNS (i.e. improved equal-average equal -variance NNS) method. Eq.6 needs two extra memories to store My_i and V_{yi} for y_i and three "±", two "×" and one "Cmp" operations for a complete rejection test.

Independently, the previous work [4] proposed a codeword rejection rule (i.e. Algorithm II in [4]) by using L_2 norm information as

$$\left[\|x\|_2 - \|y_i\|_2 \right]^2 \geq d_{\min}^2 \tag{7}$$

where $\|x\|_2 = \sqrt{\sum_{j=1}^{k} x_j^2}$ and $\|y_i\|_2 = \sqrt{\sum_{j=1}^{k} y_{i,j}^2}$ are L_2 norm of the vector x and y_i, respectively. If Eq.7 holds, then reject y_i safely. Eq.7 needs one extra memory to store $\|y_i\|_2$ for each y_i and one "±", one "×" and one "Cmp" operations for a rejection test.

In order to reduce search space more efficiently, the previous work [6] recently proposed to combine Eq.4, Eq.5 and Eq.7 to derive an EEENNS (i.e. equal-average equal-variance equal-norm NNS) method. But [6] ignored to use Eq.6, which is more powerful than Eq.4 and Eq.5.

Furthermore, it is easy to prove that the mean, the variance and L_2 norm of x have a relation like $\left[\|x\|_2 \right]^2 = k \times (Mx)^2 + (V_x)^2$ by expanding each of three features according to their definitions. For vector y_i, it is $\left[\|y_i\|_2 \right]^2 = k \times (My_i)^2 + (V_{y_i})^2$. Therefore, only *two* among these three statistical features are independent. It can be predicted that not all of three rejection tests given in Eq.4, Eq.5 and Eq.7 are necessary but two rejection tests are sufficient.

In addition to 1-dimensional statistical features of a vector, a 2-dimensional feature named as partial sum, which is constructed by dividing a vector in half, is proposed in the previous work [7], [8], [9] to represent a k-dimensional vector in a finer way. Partial sum of x and y_i is defined as

$$\begin{aligned} S_{x,1} &= \sum_{j=1}^{k/2} x_j \quad , \quad S_{x,2} = \sum_{j=k/2+1}^{k} x_j \\ S_{y_i,1} &= \sum_{j=1}^{k/2} y_{i,j} \quad , \quad S_{y_i,2} = \sum_{j=k/2+1}^{k} y_{i,j} \end{aligned} \tag{8}$$

Then the previous work [8] proposed a core codeword rejection rule by using partial sum information as

$$(S_{x,1} - S_{y_i,1})^2 + (S_{x,2} - S_{y_i,2})^2 \geq k \times d_{\min}^2 / 2 \tag{9}$$

If Eq.9 holds, then reject y_i safely. To use Eq.9 directly, it needs two extra memories to store $S_{y_i,1}$, $S_{y_i,2}$ for each y_i and three "±", two "×" and one "Cmp" operations for a rejection test when $k \times d_{\min}^2 / 2$ is stored as a whole.

3. PROPOSED METHOD

First, in order to more effectively use both the mean and the variance information, *IEENNS method given in Eq.6 instead of EENNS method given in Eq.5 should be adopted* after the rejection test by Eq.4 completed. In this case, it can be proved that the rejection test by Eq.7 that is based on L_2 norm information becomes totally redundant. A relation can be derived as (Appendix)

$$\left[\|x\|_2 - \|y_i\|_2 \right]^2 \leq k(Mx - My_i)^2 + (V_x - V_{y_i})^2 \tag{10}$$

Mathematically, Eq.10 concludes that IEENNS method of Eq.6 is definitely more powerful than L_2-norm method of Eq.7. In other words, Eq.6 can cover Eq.7 completely or it is unnecessary at all to use Eq.7 for rejection test again after Eq.6 is carried out.

Second, in order to insert an effective rejection test after IEENNS method completed, instead of L_2 norm method used in [6], *partial sum method using Eq.9 is suggested for further rejection test* because Eq.9 is not completely covered by Eq.6. Eq.9 is independent.

Because the mean and the sum have a fixed relation of "1/k", how to efficiently combine Eq.6 and Eq.9 becomes important from the computation point of view. Let $\widetilde{S}_{x,1} = S_{x,1} / k$, $\widetilde{S}_{x,2} = S_{x,2} / k$, $\widetilde{S}_{y_i,1} = S_{y_i,1} / k$, $\widetilde{S}_{y_i,2} = S_{y_i,2} / k$, Eq.9 changes to

$$(\widetilde{S}_{x,1} - \widetilde{S}_{y_i,1})^2 + (\widetilde{S}_{x,2} - \widetilde{S}_{y_i,2})^2 \geq d_{\min}^2 / (2k) \tag{11}$$

Obviously, $Mx = \widetilde{S}_{x,1} + \widetilde{S}_{x,2}$ *and* $My_i = \widetilde{S}_{y_i,1} + \widetilde{S}_{y_i,2}$ are true. Because ENNS rejection test by Eq.4 is used first and Mx, My_i are already stored, this implies that not two but only one

partial sum is independent and needs to be stored again. Suppose only the first $\widetilde{S}_{y_i,1}$ is stored for each y_i. As a result, Eq.11 becomes

$$\left(\widetilde{S}_{x,1}-\widetilde{S}_{y_i,1}\right)^2+\left[(Mx-My_i)-\left(\widetilde{S}_{x,1}-\widetilde{S}_{y_i,1}\right)\right]^2 \geq d_{\min}^2/(2k) \quad (12)$$

Because Eq.12 is always performed after Eq.4, the value of $(Mx-My_i)$ must already be available at this moment. After $\left(\widetilde{S}_{x,1}-\widetilde{S}_{y_i,1}\right)$ is computed, $\left[(Mx-My_i)-\left(\widetilde{S}_{x,1}-\widetilde{S}_{y_i,1}\right)\right]$ just needs once "±" operation. As a result, Eq.12 also needs the exact same three "±", two "×" and one "Cmp" operations for a rejection test as Eq.9 but Eq.12 can save one extra memory compared to Eq.9.

In summary, a rejection test sequence in this paper is suggested as: (1) to use ENNS method of Eq.4, (2) to use IEENNS method of Eq.6 and (3) to use the enhanced partial sum method of Eq.12. In total, three extra memories are necessary for off-line storing the mean, the variance and the first partial sum for each codeword y_i.

In contrast, concerning memory requirement, EEENNS method in the previous work [6] suggested to just store two features of the mean and L_2 norm off-line for each codeword y_i and then on-line computing its variance by using the relation $(V_{y_i})^2=\|y_i\|_2^2-k\times(My_i)^2$ when it is necessary. This way can indeed save one memory for storing V_{y_i} but it will introduce too much extra on-line computational cost. In fact, to obtain the value of V_{y_i}, it needs one "±", two "×" and one square root operation (Sqrt) for each codeword y_i that is to be tested. This part of on-line computational cost is extremely heavy. Therefore, for fast VQ encoding, it is very profitable to off-line compute the value of V_{y_i} and to use one more memory to store it. Furthermore, if one memory must be saved depending on a practical requirement, it is $\|y_i\|_2$ but not V_{y_i} should be computed on-line for each candidate y_i because in EEENNS method, Eq.5 using variance is always executed before Eq.7 using L_2 norm. If a rejection test by Eq.5 is successful, it becomes unnecessary to compute $\|y_i\|_2$ anymore so as to avoid on-line computation for it.

Concerning computational cost, because the input x is unknown before VQ encoding, all of its features must be computed on-line. Because L_2 norm feature of input x is not used in this paper, it can further save (k–1) times "±", k times "×" and one "Sqrt" for total on-line computations.

4. EXPERIMENTAL RESULTS

Simulation experiments with MATLAB are conducted. Four typical 8-bit, 512×512 standard images (Lena, F-16, Pepper and Baboon) that have very different details are used to test the effectiveness of the proposed search method. The most common 4×4 image block size is chosen by taking an appropriate trade-off into account between the compression ratio and image quality in PSNR. Codebooks of size 128, 256 and 512 are used that have been generated with Lena image as a training set by Kohonen's self-organizing map (SOM) combing with perceptual property of human vision system. EEENNS method in [6] is used as the benchmark for a comparison of search efficiency.

Concerning memory requirement, EEENNS method needs two extra memories but this work needs three extra memories for storing off-line constructed features of a vector. Because the encoding time of VQ depends on programmer's skills and computer performance strongly, it fluctuates in different implementation environments obviously. Therefore, the computational complexity instead of encoding time is adopted in this paper for performance comparison in order to achieve an invariant evaluation that has nothing to do with the executing environment. Concerning computational cost, it is evaluated by two kinds of assessments.

The first assessment is how many codewords are still remained after finishing each rejection test in a winner search process. In other words, it actually assesses how small the search space can be reduced from the whole codebook after each rejection test. A smaller reduced search space is better.

This reduced search space is the key issue to search efficiency for fast VQ encoding methods. Numbers of the remaining codewords in the reduced search spaces per input vector are summarized in Table 1.

From Table 1, it is obvious that this work can reduce much more search space than EEENNS method because the rejection test by Eq.6 is certainly more powerful than the rejection tests by separately using the mean or the variance in [6]. In addition, Eq.12 provides a new rejection test to reduce even more search space.

On the other hand, because EEENNS method is a 3-step rejection method using 1-dimensional features only but this work is a 3-step rejection method using either 1-dimensional or 2-dimensional features, they need very different computational overhead for each rejection test. In order to have a complete assessment about the overall computational cost used in both methods, the second assessment is how many additions (Add), multiplications (Mul), comparisons (Cmp) and square root (Sqrt) operations are used totally. The results are summarized in Table 2.

From Table 2, it is clear that this work can reduce total computational cost obviously compared to the latest EEENNS method [6]. This is because this work can achieve a much smaller reduced search space than EEENNS method to guarantee less Euclidean distance computations in it. Regarding the number of "Sqrt" operations, because FS method only uses squared min Euclidean distance d^2_{min} for comparison, it does not need any "Sqrt" operation. Because EEENNS method needs to on-line construct two features of the variance and L_2 norm for the input x firstly, it needs twice "Sqrt" operations; Then, EEENNS method needs to compute d_{min} in order to obtain a baseline for comparison by once "Sqrt" operation and a number of "Sqrt" operations during a search progress for updating d_{min}; Finally, because the variance of codeword is not stored off-line and must be computed on-line by using $(V_{y_i})^2=\|y_i\|_2^2-k\times(My_i)^2$, it also needs "Sqrt" operations. Thus, EEENNS method uses a lot of "Sqrt" operations as demonstrated in Table 2. In contrast, this work only needs to use once "Sqrt" operation for on-line computing the variance of the input x because the baseline for all comparisons in this work is d^2_{min} other than d_{min}.

5. CONCLUSION

In this paper, three contributions are made. First, Eq.10 is mathematically proved to show the latest EEENNS method has computational redundancy due to adopting the third feature L_2 norm of a vector for rejection test. In fact, IEENNS method is more powerful than EEENNS method. Second, concerning on-line computing the variance for candidate codeword in a search process so as to save one memory, it certainly introduces heavy extra computational burden. Because the purpose is the encoding speed, it is suggested to use one memory to store the variance for each codeword off-line. Third, in order to enhance IEENNS

method further, another 2-dimensional feature of partial sum is integrated, which is rather effective for reducing more search space. The simulation experiments confirmed that this work is more search-efficient than EEENNS method.

APPENDIX

Base on the basic inequality $a^2+b^2 \geq 2ab$, $\left[\|x\|_2\right]^2 = k \times (Mx)^2 + (V_x)^2$ and $\left[\|y_i\|_2\right]^2 = k \times (My_i)^2 + (V_{y_i})^2$, the deduction below holds, which completes the proof of Eq.10.

$$
\begin{aligned}
& k \times (Mx)^2 (V_{y_i})^2 + k \times (My_i)^2 (V_x)^2 \geq 2k \times Mx V_x My_i V_{y_i} \\
\Leftrightarrow\ & k^2 \times (Mx)^2 (My_i)^2 + k \times (Mx)^2 (V_{y_i})^2 + k \times (My_i)^2 (V_x)^2 + (V_x)^2 (V_{y_i})^2 \\
& \geq k^2 \times (Mx)^2 (My_i)^2 + 2k \times Mx V_x My_i V_{y_i} + (V_x)^2 (V_{y_i})^2 \\
\Leftrightarrow\ & \left[k \times (Mx)^2 + (V_x)^2\right]\left[k \times (My_i)^2 + (V_{y_i})^2\right] \\
& \geq \left[k \times (Mx)(My_i) + (V_x)(V_{y_i})\right]^2 \\
\Leftrightarrow\ & \left[\|x\|_2\right]^2 \left[\|y_i\|_2\right]^2 \geq \left[k \times (Mx)(My_i) + (V_x)(V_{y_i})\right]^2 \\
\Leftrightarrow\ & \left[\|x\|_2\right]\left[\|y_i\|_2\right] \geq \left[k \times (Mx)(My_i) + (V_x)(V_{y_i})\right] \\
\Leftrightarrow\ & -2 \times \left[\|x\|_2\right]\left[\|y_i\|_2\right] \leq -2k \times (Mx)(My_i) - 2 \times (V_x)(V_{y_i}) \\
\Leftrightarrow\ & \left[\|x\|_2\right]^2 - 2 \times \left[\|x\|_2\right]\left[\|y_i\|_2\right] + \left[\|y_i\|_2\right]^2 \\
& \leq \left[\|x\|_2\right]^2 - 2k \times (Mx)(My_i) - 2 \times (V_x)(V_{y_i}) + \left[\|y_i\|_2\right]^2 \\
\Leftrightarrow\ & \left[\|x\|_2 - \|y_i\|_2\right]^2 \\
& \leq \left[k \times (Mx)^2 + (V_x)^2\right] - 2k \times (Mx)(My_i) - 2 \times (V_x)(V_{y_i}) + \left[k \times (My_i)^2 + (V_{y_i})^2\right] \\
\Leftrightarrow\ & \left[\|x\|_2 - \|y_i\|_2\right]^2 \\
& \leq \left[k \times (Mx)^2 - 2k \times (Mx)(My_i) + k \times (My_i)^2\right] + \left[(V_x)^2 - 2 \times (V_x)(V_{y_i}) + (V_{y_i})^2\right] \\
\Leftrightarrow\ & \left[\|x\|_2 - \|y_i\|_2\right]^2 \leq k \times (Mx - My_i)^2 + (V_x - V_{y_i})^2
\end{aligned}
$$

6. REFERENCES

[1] N.M.Nasarabadi and R.A.King, "Image coding vector quantization: A review," *IEEE Trans. Commun.,* vol. 36, no.8, pp. 957-971, 1988.

[2] L.Guan and M.Kamel, "Equal-average hyperplane partitioning method for vector quantization of image data," *Pattern Recognition Letters*, vol.13, no.10, pp.693-699, 1992.

[3] C.H.Lee and L.H.Chen, "Fast closest codeword search algorithm for vector quantization," *IEE Proc.-Vision Image Signal Processing*, vol.141, no.3, pp.143-148, 1994.

[4] C.M.Huang, Q.Bi, G.S.Stiles and R.W.Harris, "Fast full search equivalent encoding algorithm for image compression using vector quantization," *IEEE Trans. Image Processing*, vol.1, no.7, pp.413-416, 1992.

[5] S.Baek, B.Jeon and K.Sung, "A fast encoding algorithm for vector quantization," *IEEE Signal Processing Letters*, vol.4, pp.325-327, Dec. 1997.

[6] Z.Lu and S.Sun, "Equal-average equal-variance equal-norm nearest neighbor search algorithm for vector quantization," *IEICE Trans. INF. & SEST.*, vol. E86-D, pp.660-663, 2003.

[7] Z.Pan, K.Kotani and T.Ohmi, "A hierarchical fast encoding algorithm for vector quantization with PSNR equivalent to full search," *2002 IEEE International Symposium on Circuits and Systems (ISCAS 2002)*, vol. I, pp.797-800, May 2002.

[8] Z.Pan, K.Kotani and T.Ohmi, "A fast full search equivalent encoding method for vector quantization by using appropriate features", *2003 IEEE International Conference on Multimedia and Expo (ICME 2003)*, vol. II, pp.261-264, July 2003.

[9] Z.Pan, K.Kotani and T.Ohmi, "An improved fast encoding method for vector quantization based on memory-efficient data structure," *2004 IEEE International Conference on Multimedia and Expo (ICME 2004),* June 2004.

Table 1 Comparison of the remained Euclidean distance computations per input vector after each rejection test

Method \ Reduced space		Codebook=128				Codebook=256				Codebook=512			
		Lena	F-16	Pepper	Baboon	Lena	F-16	Pepper	Baboon	Lena	F-16	Pepper	Baboon
FS	Test	128	128	128	128	256	256	256	256	512	512	512	512
EEENNS method	By mean	8.75	7.41	9.73	25.51	16.27	14.17	18.58	49.60	29.81	27.40	35.83	98.29
	By variance	4.64	4.57	4.91	19.96	7.82	7.91	8.79	37.92	13.31	13.89	16.39	72.05
	By L_2 norm	4.50	4.39	4.75	19.08	7.61	7.62	8.54	36.24	12.97	13.38	15.93	68.87
This work	By Eq.4	8.75	7.41	9.73	25.51	16.27	14.17	18.58	49.60	29.81	27.40	35.83	98.29
	By Eq.6	3.83	3.61	3.90	15.89	6.45	6.24	6.95	30.09	10.93	10.76	12.84	55.57
	By Eq.12	3.08	2.64	3.06	12.55	5.02	4.35	5.31	23.53	8.15	7.34	9.45	43.81

Table 2 Comparison of the total computational cost per input vector in terms of arithmetical operations

Method \ Total cost		Codebook=128				Codebook=256				Codebook=512			
		Lena	F-16	Pepper	Baboon	Lena	F-16	Pepper	Baboon	Lena	F-16	Pepper	Baboon
FS	Add	3968	3968	3968	3968	7936	7936	7936	7936	15872	15872	15872	15872
	Mul	2048	2048	2048	2048	4096	4096	4096	4096	8192	8192	8192	8192
	Cmp	128	128	128	128	256	256	256	256	512	512	512	512
	Sqrt	0	0	0	0	0	0	0	0	0	0	0	0
EEENNS method	Add	258.15	251.83	268.45	773.40	375.80	372.01	411.10	1388.8	580.01	588.91	689.78	2564.5
	Mul	140.28	134.93	146.14	407.39	205.99	200.67	225.30	291.75	319.53	320.12	378.66	1351.7
	Cmp	47.28	44.14	50.54	155.17	78.99	75.02	88.35	291.75	134.14	131.73	161.23	556.30
	Sqrt	13.14	11.36	14.05	30.09	21.13	18.40	23.36	54.72	34.99	32.01	40.90	103.87
This work	Add	221.33	211.39	224.51	664.54	321.24	310.22	340.73	1190.2	491.89	481.55	564.90	2140.5
	Mul	114.77	107.91	116.74	355.91	171.72	163.44	182.85	644.47	268.25	260.24	309.14	1167.7
	Cmp	26.49	24.26	27.59	76.85	42.19	39.00	45.78	141.31	68.82	65.26	79.96	262.25
	Sqrt	1	1	1	1	1	1	1	1	1	1	1	1