A MEMORY-EFFICIENT FAST ENCODING METHOD FOR VECTOR QUANTIZATION USING 2-PIXEL-MERGING SUM PYRAMID

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 05, Aramaki, Aoba-ku, Sendai, 980-8579, Japan
E-mail: pzb@fff.niche.tohoku.ac.jp

ABSTRACT

Vector quantization (VQ) is a famous signal compression method. In a framework of VQ encoding, the fast search method for finding the best-matched codeword (winner) is a key issue because it is the time bottleneck for practical applications. To speed up VQ encoding process, some fast search methods that are based on the concept of multi-resolutions by introducing a pyramid data structure have already been proposed in previous works [5]-[7]. However, there still exist two serious problems in them. First, they need a lot of extra memories for storing all purposely-constructed intermediate levels in a pyramid, which becomes an overhead of memory. Second, they completely discard the obtained Euclidean distance that has already been computed at an intermediate level whenever a rejection test fails at this level during a search process, which becomes an overhead of computation.

In order to solve the overhead problems of both memory and computation as described above, this paper proposes a memory-efficient storing way for a vector and a recursive computation way for Euclidean distance level by level based on a 2-pixel-merging (2-PM) sum pyramid, which can thoroughly reuse the obtained value of Euclidean distance at any level to compute the next rejection test condition at a successive level. Mathematically, this method does not need any extra memories at all and can reduce the original computational burden that is needed in a conventional non-recursive computation way to about half at each level. Experimental results confirmed that the proposed method outperforms the previous works obviously.

1. INTRODUCTION

In conventional VQ [1] 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

$$d^2(I, C_i) = \sum_{j=1}^{k} (I_j - C_{i,j})^2 \qquad i = 1,2,\cdots,N_c \tag{1}$$

where I is the current image block, C_i is the i^{th} codeword, j represents the j^{th} element of a vector, k (=n×n) is the vector dimension and N_c is the codebook size.

Thus, a best-matched codeword (winner) with minimum distortion can be determined straightforwardly by

$$d^2(I, C_w) = \min_i \left[d^2(I, C_i) \right] \qquad i = 1,2,\cdots,N_c \tag{2}$$

where C_w means winner and subscript "w" is the index of winner. This process for finding winner is called a full search (FS). Once "w" has been found, which uses much less bits than C_w, VQ only transmits this index "w" instead of C_w to reduce data amount for the 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 codword one by one.

Obviously, the principle of VQ encoding implies that only the sole winner C_w has to be found by an exact Euclidean distance computation but all other C_i ($i \neq w$) has to be rejected actually. This means that an exact Euclidean distance for C_i ($i \neq w$) is unnecessary. Instead, it is sufficient to just know whether Euclidean distance from I to C_i ($i \neq w$) is "larger" than the minimum Euclidean distance from I to C_w or not. In other words, VQ encoding can also be viewed as a process for rejecting all non best-matched codewords rather than finding a best-matched codeword. This property of VQ provides a possibility of roughly computing or estimating Euclidean distance to see whether it is really "larger" so as to make a codeword rejection.

Because the high dimension k of a vector is the main problem for computing Euclidean distance in VQ, it is very important to firstly use appropriate low dimensional features to approximately express a vector and then to roughly compute Euclidean distance between two vectors based on them for a rejection. One class of methods [2]-[4] is based on using the scalar statistical features of a vector (i.e. L_1 norm, the variance and L_2 norm). Another class of methods [5]-[7] is based on the multi-resolution concept by introducing an appropriate pyramid data structure to describe an original vector with a series of low dimensional intermediate levels. Both of them are very search-efficient but they suffer from the overhead problems of memory and computation. This paper aims at improving the methods proposed in [5]-[7] and completely overcoming the two overhead problems.

2. RELATED PREVIOUS WORKS

During a winner search process, suppose the "so far" minimum Euclidean distance is d_{min}. Based on statistical features of a vector, the previous work [2] proposed a rejection rule as: If $k(MI - MC_i)^2 \geq d_{min}^2$ holds, then reject C_i safely, where MI is the mean of an input image block I and MC_i means the same for C_i. This is the famous ENNS method. Then, the previous work [3] proposed a

supplementary rejection rule when ENNS method fails as: If $k(MI - MC_i)^2 + (VI - VC_i)^2 \geq d^2_{\min}$ 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 EENNS method. To improve [3] further, the previous work [4] proposed another additional codeword rejection rule when EENNS method also fails as: If $(L_2 I - L_2 C_i)^2 \geq d^2_{\min}$ holds, then reject C_i safely as well, where L_2I is the L_2 norm of I and L_2C_i means the same for C_i. This is EEENNS method (i.e. equal-average equal-variance equal-norm nearest neighbor search). EEENNS method is the current fastest search method by using statistical features of a vector. But EEENNS method cannot avoid the overhead of three extra memories for storing the mean, the variance and L_2 norm of each C_i. Furthermore, if any one of these three rejection tests in EEENNS method fails, the overhead of computation inevitably occurs as well.

On the other hand, in order to realize a multi-resolution description for a vector, a pyramid data structure can be taken into account naturally. Then, a 4-PM mean pyramid as shown in Fig.1 (a) is proposed in the previous works [5], [6].

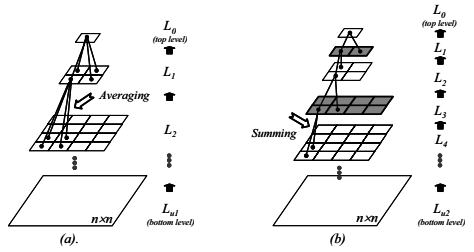

Fig.1. For an n×n block, (a) a 4-PM mean pyramid with bottom level $u1=\log_4(n\times n)$ and (b) a 2-PM sum pyramid with bottom level $u2=\log_2(n\times n)=2\times u1$. One new level is sandwiched in-between every 2 levels of a conventional 4-PM pyramid (shadowed). All levels above the bottom level are called as intermediate levels afterwards.

Then a hierarchical rejection rule is set up in [5], [6] as

$$d^2_{m,u1}(I,C_i) \geq \cdots \geq 4^{u1-v} d^2_{m,v}(I,C_i) \geq \cdots \geq 4^{u1-1} d^2_{m,1}(I,C_i) \geq 4^{u1} d^2_{m,0}(I,C_i) \quad (3)$$

Suppose $MI_{v,m}$ is the m^{th} pixel (the mean after roundoff) at the v^{th} level in a 4-PM mean pyramid for I and $MC_{i,v,m}$ implies the same thing to C_i for $m\in[1\sim4^v]$, then Euclidean distance at the v^{th} level for $v\in[0\sim u1]$ between the two 4-PM mean pyramids is $d^2_{m,v}(I,C_i) \overset{Def}{=} \sum_{m=1}^{4^v}(MI_{v,m} - MC_{i,v,m})^2$, where the real Euclidean distance at the bottom level L_{u1} can be obtained by $d^2(I, C_i)=d^2_{m,u1}(I, C_i)$. For a 4×4 block, u1=2.

Thus, at any v^{th} level for $v\in[0\sim u1]$, if $4^{u1-v}d^2_{m,v}(I, C_i) > d^2_{\min}$ holds, then the current real Euclidean distance $d^2_{m,u1}(I, C_i)$ is definitely larger than $d^2_{\min}$. As a result, the search can be terminated at this v^{th} level and C_i can be rejected safely.

Originally, this 4-PM mean pyramid is used for a progressive image transmission starting from the top level. If the transmission can be terminated before it reaches the bottom L_{u1} level by a practical requirement at the receiver, image compression can be realized because the transmitted maximum data amount so far will be $(1+4^1+4^2+\cdots+4^{u1-1})=(4^{u1}-1)/3=(k-1)/3$. Obviously, two constraints exist in constructing a pyramid for image transmission. First, it must guarantee each pixel value is an integer and in [0, 255] interval for a 8-bit displaying. Second, it must keep the aspect ratio of a 2-dimensional image to be constant in order to show it correctly. Therefore, the averaging, the roundoff operation and 2×2 pixel merging way are required.

This 4-PM mean pyramid method needs $(1+4^1+4^2+\cdots+4^{u1-1})$ extra memories to store all intermediate levels for each C_i. it is much more than the overhead of memory in EEENNS method. Obviously, if a rejection test fails at a certain level, the overhead of computation will inevitably occurs as well. This is because when a test fails at the v^{th} level, the obtained Euclidean distance $4^{u1-v}d^2_{m,v}(I, C_i)$ is discarded completely and then $4^{u1-(v+1)}d^2_{m,v+1}(I, C_i)$ is computed again for next rejection test at the $(v+1)^{th}$ level. This is surly a waste of computation.

To use multi-resolution concept more effectively, it is surely better to construct more levels in a pyramid. In order to improve the 4-PM mean pyramid, a 2-PM sum pyramid [7] is proposed as shown in Fig.1 (b), which can completely remove away the two constraints for constructing a pyramid for image transmission and double the resolutions in a pyramid. However, the final distortion measurement adopted in [7] is Manhattan distance. This paper tries to exploit the power of the 2-PM sum pyramid proposed in [7] further by introducing Euclidean distance and meanwhile completely avoiding the overhead of memory and computation.

3. PROPOSED METHOD

The advantages of directly using a 2-PM sum pyramid compared to a 4-PM mean pyramid are: (1) exact for operations in integer form; (2) doubled number of intermediate levels for an easier rejection; (3) lighter on-line computational burden for constructing the pyramid of an input I. Accordingly, its disadvantages are: (1) more extra memories; (2) probably more overhead of computation.

The first two advantages are obvious. An analysis on the third advantage is given below. It is lighter to construct a 2-PM sum pyramid because it needs $(2^{u2-1}+\cdots+2^2+2^1+1)=2^{u2}-1=n\times n-1$ addition (±). In contrast, a 4-PM mean pyramid needs $(4-1)\times(4^{u1-1}+\cdots+4^2+4^1+1)=4^{u1}-1=n\times n-1$ addition (±), $(4^{u1-1}+\cdots+4^2+4^1+1)=(n\times n-1)/3$ division (÷) for averaging and $(4^{u1-1}+\cdots+4^2+4^1+1)=(n\times n-1)/3$ truncating operations for converting the mean from a real to an integer. Regarding disadvantage of memory, it needs $(2^{u2-1}+\cdots+2^2+2^1+1)=2^{u2}-1=n\times n-1$ extra memories for a 2-PM sum pyramid but just $(4^{u1-1}+\cdots+4^2+4^1+1)=(n\times n-1)/3$ extra memories for a 4-PM mean pyramid. In addition, a 2-PM sum pyramid has to use a longer [8+(u2–v)] bit memory unit at the v^{th} level for $v\in[0\sim u2]$ while a 4-PM mean pyramid always uses 8-bit one for all levels. But this will not cause any problem for a 32-bit general-purpose computer.

Similar to Eq.3, a new hierarchical rejection rule can be got using Euclidean distance and a 2-PM sum pyramid as

$$d^2_{s,u2}(I,C_i) \geq \cdots \geq 2^{-(u2-v)} d^2_{s,v}(I,C_i) \geq \cdots \geq 2^{-(u2-1)} d^2_{s,1}(I,C_i) \geq 2^{-u2} d^2_{s,0}(I,C_i) \quad (4)$$

where Euclidean distance at the v^{th} level for $v\in[0\sim u2]$ is $d^2_{s,v}(I,C_i) \overset{Def}{=} \sum_{m=1}^{2^v}(SI_{v,m} - SC_{i,v,m})^2$, $SI_{v,m}$ is the m^{th} pixel at the v^{th} level for I and $SC_{i,v,m}$ means the same to C_i for $m\in[1\sim2^v]$.

The real Euclidean distance at the bottom level L_{u2} can be obtained by $d^2(I, C_i)=d^2_{s,u2}(I, C_i)$. For a 4×4 block, u2=4.

Thus, at any v^{th} level for v∈[0~u2], if $2^{-(u2-v)}d^2_{s,v}(I, C_i) > d^2_{min}$ holds, then the current real Euclidean distance $d^2_{s,u2}(I, C_i)$ is definitely larger than d^2_{min}. As a result, the search can be terminated at this v^{th} level and C_i can be rejected safely.

Proof of Eq.4

From a 2-PM sum pyramid shown in Fig.1 (b), it is clear that $SI_{v,m} = SI_{v+1,2m-1} + SI_{v+1,2m}$ and $SC_{i,v,m} = SC_{i,v+1,2m-1} + SC_{i,v+1,2m}$ are true for m∈[1~2^v]. By using $(a^2+b^2)\geq(a+b)^2/2$, we have

$$\begin{aligned}
d^2_{s,v+1}(I, C_i) &= \sum_{m=1}^{2^{v+1}} \left(SI_{v+1,m} - SC_{i,v+1,m}\right)^2 \\
&= \sum_{m=1}^{2^v} \left[\left(SI_{v+1,2m-1} - SC_{i,v+1,2m-1}\right)^2 + \left(SI_{v+1,2m} - SC_{i,v+1,2m}\right)^2\right] \\
&\geq 2^{-1}\sum_{m=1}^{2^v} \left[\left(SI_{v+1,2m-1} - SC_{i,v+1,2m-1}\right) + \left(SI_{v+1,2m} - SC_{i,v+1,2m}\right)\right]^2 \\
&= 2^{-1}\sum_{m=1}^{2^v} \left[\left(SI_{v+1,2m-1} + SI_{v+1,2m}\right) - \left(SC_{i,v+1,2m-1} + SC_{i,v+1,2m}\right)\right]^2 \\
&= 2^{-1}\sum_{m=1}^{2^v} \left[SI_{v,m} - SC_{i,v,m}\right]^2 \\
&= 2^{-1} d^2_{s,v}(I, C_i)
\end{aligned}$$

Because the initial distance computation is $d^2_{s,0}(I, C_i)$ and totally (u2+1) levels are available, Eq.4 can be easily achieved by using the relation $d^2_{s,v+1}(I, C_i)\geq 2^{-1}d^2_{s,v}(I, C_i)$.

Comparing Eq.4 with Eq.3, it is obvious that Eq.4 has doubled the number of rejection tests for a codeword. If Eq.4 is directly used, it also needs (n×n−1) extra memories for each C_i. And when a test at the v^{th} level fails, the obtained $d^2_{s,v}(I, C_i)$ is discarded completely and $d^2_{s,v+1}(I, C_i)$ must be computed again. Eq.4 also has two overhead problems.

Therefore, we firstly discuss how to store a 2-PM sum pyramid in a memory-efficient way. In fact, it is unnecessary to store all pixels at each level because a lot of redundancies exist in them. From $SI_{v,m} = SI_{v+1,2m-1} + SI_{v+1,2m}$, it achieves $SI_{v+1,2m} = SI_{v,m} - SI_{v+1,2m-1}$. This means that an even term is not independent and it can be obtained on-line by using previous $SI_{v,m}$ and a odd term to avoid a storage. For C_i, it is the same. Thus, only the odd terms at the v^{th} level for v∈[0~u2] must be stored. Then, $d^2_{s,v+1}(I, C_i)$ becomes

$$\begin{aligned}
d^2_{s,v+1}(I, C_i) &\overset{Def}{=} \sum_{m=1}^{2^{v+1}} \left(SI_{v+1,m} - SC_{i,v+1,m}\right)^2 \\
&= \sum_{m=1}^{2^v} \left[\left(SI_{v+1,2m-1} - SC_{i,v+1,2m-1}\right)^2 + \left(SI_{v+1,2m} - SC_{i,v+1,2m}\right)^2\right] \\
&= \sum_{m=1}^{2^v} \left\{\left(SI_{v+1,2m-1} - SC_{i,v+1,2m-1}\right)^2 + \left[\left(SI_{v,m} - SI_{v+1,2m-1}\right) - \left(SC_{i,v,m} - SC_{i,v+1,2m-1}\right)\right]^2\right\} \\
&= \sum_{m=1}^{2^v} \left\{\left(SI_{v+1,2m-1} - SC_{i,v+1,2m-1}\right)^2 + \left[\left(SI_{v,m} - SC_{i,v,m}\right) - \left(SI_{v+1,2m-1} - SC_{i,v+1,2m-1}\right)\right]^2\right\}
\end{aligned} \tag{5}$$

Based on Eq.5, it is only necessary to store $SI_{0,1}, SI_{1,1}, \left(SI_{2,1}, SI_{2,3}\right)$ $\left(SI_{u2,1}, SI_{u2,3}, \cdots, SI_{u2,2^{u2}-1}\right)$ for I. Therefore, $(1+2^0+2^1+\ldots+2^{u2-1})$ $=2^{u2}$=n×n memories in total are sufficient for storing a 2-PM sum pyramid of input I. It can save (n×n−1) extra memories needed in a direct storing way. For C_i, it is the same. This result actually reflects the requirement of information theory that states n×n memories should be sufficient for storing n×n dependent elements of a vector. Regarding computational burden using Eq.5, because all $\left(SI_{v,m} - SC_{i,v,m}\right)$must be known already for m∈[1~2^v] after the $d^2_{s,v}(I, C_i)$ check failed and the $\left(SI_{v+1,2m-1} - SC_{i,v+1,2m-1}\right)$ is computed first, Eq.5 only needs $(3\times2^v)+(2^v-1)=2^{v+2}-1$ addition (±), $2\times2^v=2^{v+1}$ multiplication (×) operations. In contrast, a direct computation way using the definition of $d^2_{s,v+1}(I, C_i)$ needs $(2^{v+1})+(2^{v+1}-1) =2^{v+2}-1$ additions (±) and 2^{v+1} multiplications (×). Obviously, Eq.5 does not introduce any extra computational burden.

Secondly, we discuss how to reuse the obtained $d^2_{s,v}(I, C_i)$ to compute $d^2_{s,v+1}(I, C_i)$ recursively in order to reduce more computational burden further. A recursive computation way is proposed by using $(a^2+b^2)=(a+b)^2-2ab$ as

$$\begin{aligned}
d^2_{s,v+1}(I, C_i) &\overset{Def}{=} \sum_{m=1}^{2^{v+1}} \left(SI_{v+1,m} - SC_{i,v+1,m}\right)^2 \\
&= \sum_{m=1}^{2^v} \left[\left(SI_{v+1,2m-1} - SC_{i,v+1,2m-1}\right)^2 + \left(SI_{v+1,2m} - SC_{i,v+1,2m}\right)^2\right] \\
&= \sum_{m=1}^{2^v} \left[\left(SI_{v+1,2m-1} - SC_{i,v+1,2m-1}\right) + \left(SI_{v+1,2m} - SC_{i,v+1,2m}\right)\right]^2 \\
&\quad - 2\times\sum_{m=1}^{2^v} \left[\left(SI_{v+1,2m-1} - SC_{i,v+1,2m-1}\right)\left(SI_{v+1,2m} - SC_{i,v+1,2m}\right)\right] \\
&= \sum_{m=1}^{2^v} \left[SI_{v,m} - SC_{i,v,m}\right]^2 \\
&\quad - 2\times\sum_{m=1}^{2^v} \left[\left(SI_{v+1,2m-1} - SC_{i,v+1,2m-1}\right)\left(SI_{v+1,2m} - SC_{i,v+1,2m}\right)\right] \\
&= d^2_{s,v}(I, C_i) - 2\times\sum_{m=1}^{2^v} \left\{\left(SI_{v+1,2m-1} - SC_{i,v+1,2m-1}\right)\times \right. \\
&\qquad \left.\left[\left(SI_{v,m} - SC_{i,v,m}\right) - \left(SI_{v+1,2m-1} - SC_{i,v+1,2m-1}\right)\right]\right\}
\end{aligned} \tag{6}$$

From Eq.6 it is obvious that $d^2_{s,v}(I, C_i)$ is completely reused again so that $d^2_{s,v+1}(I, C_i)$ can be computed recursively. Eq.6 is actually a realization of the multi-resolution concept, which implies that any higher resolution can be obtained by just adding some improvements recursively into a lower resolution rather than computing it from the very beginning once again. Eq.6 only needs $[2\times2^v+(2^v-1)+2] =(3/4)\times2^{v+2}+1$ addition (±) (Note: the factor "2" in Eq.6 is realized by 1 (±) instead of 1 (×) operation) and $2^v=(1/2)\times2^{v+1}$ multiplication (×) operations. And the initial check at L_0 level needs 1 (±) and 1 (×) operation. Because a multiplication (×) is heaviest, Eq.6 can save about half of the computational burden further compared to Eq.5 or a direct way of computing $d^2_{s,v+1}(I, C_i)$.

Based on the discussions above, a search flow can be summarized as follows: (1) Construct an accompanying 2-PM sum pyramid for each C_i off-line. Only store the odd terms but the even terms for v∈[0~u2] levels (2) Sort all codewords by the real sum at L_0 level in ascending order off-line. (3) For an input I, construct its accompanying 2-PM sum pyramid on-line. Similarly, only store the odd terms but the even terms for v∈[0~u2] levels. (4) Find an initial nearest (NN) codeword C_N among the sorted codewords by using a binary search, which is the closest codeword in terms of real sum difference $d_{s,0}(I, C_N) = \left|SI_{0,1} - SC_{N,0,1}\right|$ being minimum. It needs $\log_2(N_c)$ times comparisons (CMP). Then compute and temporally store "so far" $d^2_{min} = d^2(I, C_N)$, $d^2_{v,\,min}= 2^{u2-v}\times d^2_{min}$ in order to simplify a future rejection test at the v^{th} level for v∈[0~u2−1]. This step needs (2k−1) additions (±) and (k+u2) multiplications (×) operations. (5.1) Continue the winner search up and down around C_N one by one. Once $\left(SI_{0,1} - SC_{i,0,1}\right)^2 \geq d^2_{0,min}$ holds, terminate search for the upper part of sorted codebook when i<N or the lower part when i>N; If winner search in both upper and lower directions has been terminated, search is complete. Clearly, the current "so far" best-matched codeword must be the winner. Then search flow returns to Step 3 for encoding another new input. (5.2) Otherwise, test whether $d^2_{s,v}(I, C_i)\geq d^2_{v,min}$ is true or not for v∈[1~u2]. If it is true, reject C_i safely. (Note, Eq.6 must be introduced here for computing $d^2_{s,v}(I, C_i)$ recursively.) (6) If all tests fail for a rejection, it implies that current C_i is a

better-matched codeword, then update d^2_{min} by $d^2_{s,u2}(I, C_i)$ and all $d^2_{v,min}$ for $v \in [0 \sim u2-1]$ again. Meanwhile, update the corresponding winner index "so far". Then, return to Step (5.1) to test next codeword.

4. EXPERIMENTAL RESULTS

To compare the search performance with previous works, simulation experiments using MATLAB are conducted. Codebooks of size 256, 512 and 1024 are generated using 512×512, 8-bit Lena image as a training set based on [8]. Block size is 4×4. Search efficiency is evaluated by total computational burden in terms of the number of addition (±), multiplication (×) and comparison (CMP) operations per input vector, which consists of (1) finding the initial best-matched codeword C_N and computing the initial d^2_{min}, $d^2_{v,min}$; (2) computing test condition for a possible rejection using each statistical feature in EEENNS method or each intermediate level in a pyramid method; (3) computing the real Euclidean distance and updating "so far" d^2_{min}, $d^2_{v,min}$ again if current codeword is a better-matched one.

TABLE 1
COMPARISON OF TOTAL COMPUTATIONAL BURDEN PER INPUT VECTOR

Size	Method	Operation	Lena	F-16	Pepper	Baboon
256	Full search	Add	7936	7936	7936	7936
		Mul	4096	4096	4096	4096
		CMP	256	256	256	256
	EEENNS	Add	514.0	494.3	591.7	1706.0
		Mul	290.0	277.7	333.4	953.5
		CMP	73.1	68.1	82.2	208.0
	4-PM man pyramid	Add	424.0	353.0	463.0	1199.5
		Mul	235.6	197.2	257.8	664.9
		CMP	54.2	48.3	59.5	137.3
	2-PM sum pyramid	Add	358.5	297.8	392.0	1038.9
		Mul	125.1	105.5	136.4	347.5
		CMP	66.0	58.0	72.7	179.2
512	Full search	Add	15872	15872	15872	15872
		Mul	8192	8192	8192	8192
		CMP	512	512	512	512
	EEENNS	Add	896.0	909.2	1104.5	3338.4
		Mul	505.3	510.3	621.6	1864.9
		CMP	122.6	119.3	146.7	399.4
	4-PM mean pyramid	Add	682.5	606.3	807.6	2283.9
		Mul	380.8	339.4	450.6	1267.0
		CMP	87.2	80.5	101.7	258.1
	2-PM sum pyramid	Add	556.9	497.2	661.9	1968.0
		Mul	191.7	172.2	226.4	652.6
		CMP	106.1	97.8	125.1	340.0
1024	Full search	Add	31744	31744	31744	31744
		Mul	16384	16384	16384	16384
		CMP	1024	1024	1024	1024
	EEENNS	Add	1425.5	1670.0	1985.9	6372.7
		Mul	806.9	937.5	1120.0	3560.4
		CMP	197.3	214.0	263.2	757.0
	4-PM mean pyramid	Add	1032.5	1038.7	1389.0	4099.4
		Mul	580.9	584.5	779.5	2283.9
		CMP	138.2	139.3	179.0	477.4
	2-PM sum pyramid	Add	820.3	826.0	1104.8	3450.2
		Mul	282.1	283.7	376.3	1144.4
		CMP	166.2	168.7	218.4	623.0

From TABLE 1, it is obvious that the total computational burden by using a 2-PM sum pyramid is much less compared to other previous works, especially the number of multiplication (×) operations can be reduced greatly. This benefit mainly comes from recursively computing $d^2_{s,v}(I, C_i)$ for $v \in [1 \sim u2]$ by using Eq.6.

5. CONCLUSIONS

In this paper, a memory-efficient storing way and a recursive computation way for Euclidean distance at each level are proposed based on a 2-PM sum pyramid. Two advantages are achieved. First, memory redundancy in a 2-PM sum pyramid is completely removed by just storing the odd terms at each level. As a result, n×n memories are sufficient in total for exactly representing an n×n dimensional block. Second, total computational burden is reduced to about half by recursively computing Euclidean distance at each level compared to directly computing it according to its definition. As a result, it can completely avoid any waste to the obtained Euclidean distance at previous levels. This is in fact a physical realization of the multi-resolution concept by using a 2-PM sum pyramid. Meanwhile, It is also interesting to see that a 4-PM mean pyramid cannot benefit from a recursive computation because unlike the formula $a^2+b^2=(a+b)^2-2ab$ that can save once multiplication (×) when the value of $(a+b)^2$ term is known, the formula $a^2+b^2+c^2+d^2=(a+b+c+d)^2-2ab-2ac-2ad-2bc-2bd-2cd$ cannot reduce computational burden at all even though the value of $(a+b+c+d)^2$ term is known. The proposed method is very promising and practical to fast VQ encoding.

6. REFERENCES

[1] N.M.Nasarabadi and R.A.King, "Image coding using vector quantization: A review," IEEE Trans. Commun., vol. 36, pp.957-971, Aug. 1988.

[2] L.Guan and M.Kamel, "Equal-average hyperplane partitioning method for vector quantization of image data," Pattern Recognition Letters, vol.13, pp.693-699, Oct. 1992.

[3] 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.

[4] Z.M.Lu and S.H.Sun, "Equal-average equal-variance equal-norm nearest neighbor search algorithm for vector quantization," IEICE Trans. Information and System, vol.E86-D, pp.660-663, March 2003.

[5] C.H.Lee and L.H.Chen, "A fast search algorithm for vector quantization using mean pyramids of codewords," IEEE Trans. Commun., vol.43, pp.1697-1702, Feb. 1995.

[6] S.J.Lin, K.L.Chung and L.C.Chang, "An improved search algorithm for vector quantization using mean pyramid structure," Pattern Recognition Letters, vol.22, pp.373-379, 2001.

[7] Z.Pan, K.Kotani and T.Ohmi, "A fast encoding method for vector quantization based on 2-pixel-merging sum pyramid data structure," IEICE Trans. Fundamentals, vol.E86-A, pp.2419-2423, Sep. 2003.

[8] T.Nozawa, M.Konda, M.Fujibayashi, M.Imai, K.Kotani, S.Sugawa and T.Ohmi, "A parallel vector-quantization processor eliminating redundant calculations for real-time motion picture compression," IEEE J. Solid-State Circuits, vol.35, pp.1744-1751, Nov. 2000.