IMPROVED FAST SEARCH METHOD FOR VECTOR QUANTIZATION USING DISCRETE WALSH TRANSFORM

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

In a framework of vector quantization (VQ), 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 projection axes or Walsh transform have already been proposed in previous works [4]-[9]. However, there still exist two serious problems in them because they use both spatial domain and partial Walsh domain simultaneously. First, they need extra memories for storing projected values on selected projection axes or the first several elements in partial Walsh domain, which becomes an overhead of memory. Second, once all rejection tests fail finally, they completely discard the obtained distortion that has already been computed in partial Walsh domain and return to spatial domain to compute real Euclidean distance again from the very beginning, which is certainly a waste and becomes an overhead of computation.

In order to solve the overhead problems of both memory and computation as described above, firstly a memory-efficient storing way for a vector is proposed by completely mapping a vector into Walsh domain but NOT using the original spatial domain anymore, which can avoid extra memory requirement. Secondly, the discarded distortion in partial Walsh domain is reused so as to avoid any waste to the executed computation. In addition, a more efficient rejection test is suggested to reduce more search space. Experimental results confirmed that the proposed method outperforms the previous works obviously.

1. INTRODUCTION

In a 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 conducted block by block sequentially. The real distortion between an input image block and a codeword is a difference vector $D_i = x - y_i \quad i = 1,2,\cdots,N_c$, where $x=(x_1, x_2, \ldots, x_k)^T$ is an input, $y_i=(y_{i,1}, y_{i,2}, \ldots, y_{i,k})^T$ is the i^{th} codeword in the codebook $Y=\{y_i \mid i=1, 2, \ldots, N_c\}$, k (=n×n) is the vector dimension and "T" refers to a transpose. For simplicity, D_i is usually measured by squared Euclidean distance or its energy as

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

where j is the dimension of a vector, N_c is the codebook size.

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

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

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

Obviously, the principle of VQ encoding implies that only the sole winner y_w has to be found by an exact Euclidean distance computation but all other y_i ($i \neq w$) has to be rejected actually. This means that an exact Euclidean distance for each y_i ($i \neq w$) is redundant. Instead, it is sufficient to just know whether Euclidean distance from x to y_i ($i \neq w$) is "larger" than the minimum Euclidean distance from x to y_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 estimating Euclidean distance by a rough but lighter computation to see whether it is really "large" enough for rejecting a candidate codeword.

According to Eq.1, it is certainly profitable to obtain the error energy of D_i quicker. Because an orthogonal transform has the property of energy compaction that makes the energy of a vector compact to the first several elements, it can make the decision of "whether current distortion is large enough for a rejection" easier so as to speed up the search process. This paper aims at improving the methods proposed in [4]-[9] and completely overcoming the two overhead problems.

2. DISCRETE WALSH TRANSFORM

Many orthogonal transforms (Karhunen-Loeve transform, Haar transform, Slant transform, DCT, and Walsh transform) can be applied to signal compression by exploiting their energy compaction property. Their performances are just slightly different but their computational complexities are very different [2]. For the purpose of fast search, discrete Walsh transform is adopted in this paper because it only needs to use lighter addition (±) but heavier multiplication (×) operations. In addition, its fast algorithm is available as well [3]. As an example, the kernel of discrete Walsh transform for k=16 is shown as below

$$W_{16} = \frac{1}{4} \times \begin{bmatrix} 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&-1&-1&-1&-1&-1&-1&-1&-1 \\ 1&1&1&1&-1&-1&-1&-1&-1&-1&-1&-1&1&1&1&1 \\ 1&1&1&1&-1&-1&-1&-1&1&1&1&1&-1&-1&-1&-1 \\ 1&1&-1&-1&-1&-1&1&1&1&1&-1&-1&-1&-1&1&1 \\ 1&1&-1&-1&-1&-1&1&1&-1&-1&1&1&1&1&-1&-1 \\ 1&1&-1&-1&1&1&-1&-1&-1&-1&1&1&-1&-1&1&1 \\ 1&1&-1&-1&1&1&-1&-1&1&1&-1&-1&1&1&-1&-1 \\ 1&-1&-1&1&1&-1&-1&1&1&-1&-1&1&1&-1&-1&1 \\ 1&-1&-1&1&1&-1&-1&1&-1&1&1&-1&-1&1&1&-1 \\ 1&-1&-1&1&-1&1&1&-1&-1&1&1&-1&1&-1&-1&1 \\ 1&-1&-1&1&-1&1&1&-1&1&-1&-1&1&-1&1&1&-1 \\ 1&-1&-1&1&1&-1&-1&1&-1&1&1&-1&-1&1&1&-1 \\ 1&-1&1&-1&-1&1&-1&1&-1&1&1&-1&1&-1&-1&1 \\ 1&-1&1&-1&-1&1&-1&1&1&-1&-1&1&-1&1&-1&1 \end{bmatrix} \begin{matrix} \textit{Sequency} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{matrix} = \begin{bmatrix} W_{16,1} \\ W_{16,2} \\ W_{16,3} \\ W_{16,4} \\ W_{16,5} \\ W_{16,6} \\ W_{16,7} \\ W_{16,8} \\ W_{16,9} \\ W_{16,10} \\ W_{16,11} \\ W_{16,12} \\ W_{16,13} \\ W_{16,14} \\ W_{16,15} \\ W_{16,16} \end{bmatrix}$$

In a row of W_{16}, the number of zero crossings is defined as its *sequency*, which is an equivalent of frequency in Walsh domain. There are several ways to arrange the Walsh transform kernel by the sequency but a sequency-ordered one is more applicable to signal processing because of its physical meaning [3]. Obviously, $W_{16}W_{16}^T = I_{16}$ is true, where I_{16} is an 16×16 identity matrix. This is the unitary property of Walsh transform. For VQ encoding, suppose $z=W_{16}x$ and $w_i=W_{16}y_i$ are the corresponding Walsh transformed vectors of x and y_i, then $d^2(z, w_i)=d^2(x, y_i)$ holds. This is the energy conservation property of Walsh transform. It is the key point in this paper because the problem for finding the winner in spatial domain can be converted to find the winner in Walsh domain equivalently.

3. RELATED PREVIOUS WORK

During a winner search process, suppose the "so far" minimum Euclidean distance is d_{min}. Based on the statistical features of a vector, the previous work [4] proposed a codeword rejection rule as

$$k(Mx - My_i)^2 \geq d_{\min}^2 \tag{3}$$

If Eq.3 holds, then reject y_i safely, where Mx is the mean of an input image block x and My_i means the same for y_i. This is the famous ENNS (i.e. equal-average nearest neighbor search) method. Eq.3 needs one extra memory to store My_i for each y_i and one "±", one "×" (Eq.3 practically tests $\left(\sqrt{k}Mx - \sqrt{k}My_i\right)^2 \geq d_{\min}^2$ by off-line storing "$\sqrt{k}\times My_i$" as a whole and on-line computing "$\sqrt{k}\times Mx$" only once) and one "cmp" (comparison) operations for a rejection test. On the other hand, since $W_{k,1}=(1/\sqrt{k})\times[1,1,\ldots,1]$, the first element of x in Walsh domain is $z_1=W_{k,1}x=\sqrt{k}\times Mx$. For y_i, $w_{i,1}=W_{k,1}y_i=\sqrt{k}\times My_i$. Based on the famous partial distortion search (PDS) [5] method, it is obvious that Eq.3 is equivalent to $(z_1 - w_{i,1})^2 \geq d_{\min}^2$ in Walsh domain. It concludes that Eq.3 is actually a rejection test on the first basis axis $W_{k,1}$.

In order to improve [4], the previous work [6] sequentially used the first three basis axes of Walsh transform for rejection tests as

$$|z_1 - w_{i,1}| \geq d_{\min} \quad or \quad |z_2 - w_{i,2}| \geq d_{\min} \quad or \quad |z_3 - w_{i,3}| \geq d_{\min} \tag{4}$$

where z_2, z_3 are the second and third elements of x in Walsh transform, $w_{i,2}$, $w_{i,3}$ imply the same thing for y_i. This is DHSS (i.e. dynamical hyperplanes shrinking search) method. Why the first three basis axes are sufficient is experimentally confirmed in [6]. Eq.4 needs three extra memories to store $w_{i,1}$, $w_{i,2}$ and $w_{i,3}$ for each y_i and three "±" and six "cmp" operations at maximum for three rejection tests.

In contrast, the previous work [7] used the projection concept other than Walsh transform to give a rejection condition as

$$k(p_x - p_{y_i})^2 \geq d_{\min}^2 \tag{5}$$

where $p=[p_1, p_2, \ldots, p_k]^T$ is a projection axis, $p_x = \sum_{j=1}^{k} p_j x_i / k$ $p_{y_i} = \sum_{j=1}^{k} p_j y_{i,j} / k$ is the projected value of x and y_i onto p, respectively. If three projection axes are selected from the generalized central axis $[\pm1, \pm1, \ldots, \pm1]^T$ as defined in [7], then Eq.5 is equivalent to Eq.4 exactly. Obviously, a generalized central axis is physically a basis axis of Walsh transform with a coefficient $(1/\sqrt{k})$. Eq.5 also needs three extra memories to store three projected values p_{y_i} for each y_i and three "±", three "×" and three "cmp" operations at maximum for three rejection tests. Unlike Eq.4, Eq.5 does not use square root (√) operation for obtaining d_{min}.

Mathematically, to project a vector onto an axis leads to two components, which are a projection component and its *orthogonal component* (i.e. the orthogonal decomposition of a vector onto an projection axis). However, Eq.3, Eq.4 and Eq.5 only used the projection component of a vector but completely ignored its orthogonal component. In order to include the orthogonal component into a rejection test, the previous work [8] proposed a rejection rule as

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

where $V_x = d(x, L_x)$ and $L_x = [Mx, Mx, \cdots, Mx]^T$. Geometrically, L_x is the projection point of vector x onto the central axis $p=[1,1,\ldots,1]^T$ as defined in [8]. For y_i, it is the same. This is ENNSV (i.e. equal-average nearest neighbor search with variance) method. Clearly, V_x is the orthogonal component of x if vector x is decomposed using the central axis. Besides, it is interesting to notice that V_x is just the variance among all elements of vector x. Eq.6 needs two extra memories to store My_i and V_{yi} for each y_i and three "±", two "×" and one "cmp" operations for a rejection test.

From Eq.3 to Eq.6, it is apparent that although multiple axes (either basis axis of Walsh transform or generalized central axis) are used, they are used on by one or separately. Because $d^2(x, y_i)$ reflects error energy, it is surely better to use multi axes *together*. From this point of view, the previous work [9] proposed a rejection rule as

$$(z_1 - w_{i,1})^2 + (z_2 - w_{i,2})^2 + (z_3 - w_{i,3})^2 \geq d_{\min}^2 \tag{7}$$

Eq.7 needs three extra memories to store $w_{i,1}$, $w_{i,2}$ and $w_{i,3}$ for each y_i. In practice, Eq.7 uses an incremental search that means to test $(z_1 - w_{i,1})^2 \geq d_{\min}^2$ and $(z_1 - w_{i,1})^2 + (z_2 - w_{i,2})^2 \geq d_{\min}^2$ first in order to avoid possible computational overhead. The reason is that many codewords can be rejected by these first two tests. Eq.7 needs five "±", three "×" and three "cmp" operations at maximum for a complete rejection test. Obviously, Eq.7 is more powerful than Eq.3, Eq.4 and Eq.5.

In fact, all of the previous works [4]~[9] used both spatial domain and *partial* Walsh domain. Here "partial" implies that not all but just a few basis axes of Walsh transform are used for rejection tests while final real Euclidean distance must be computed in spatial domain. Thus, they have to store all information in spatial domain (i.e. k memories for storing a codeword y_i). At the same time, they have to use some extra memories to store the mean My_i or the projection component $w_{i,1}$, $w_{i,2}$ and $w_{i,3}$ or the variance V_{y_i} for each codeword y_i as its features that are to be used only for rejection tests but real Euclidean distance computation. The rejection tests can be virtually viewed as information processing in partial Walsh domain because they use only a few basis axes of Walsh transform inexplicitly. Once all rejection tests ultimately fail in partial Walsh domain, the search must return to spatial domain to compute real Euclidean distance from the very beginning. In this case, all information obtained in partial Walsh domain cannot be reused directly and are discarded. It is obviously a waste.

4. PROPOSED METHOD

In order to overcome both memory and computation overhead that come from using both spatial domain and partial Walsh domain, this paper propose to use Walsh

domain *only* but not original spatial domain anymore for winner search. This is possible because $d^2(z, w_i)=d^2(x, y_i)$ always holds. This paper aims at improving the method given in the previous work [6] (i.e. Eq.4) and [9] (i.e. Eq.7).

Concerning the memory requirement, because it is only necessary to store w_i but not y_i anymore in this paper, it is clear that k memories are sufficient for each codeword w_i. As a result, memory overhead can be solved completely.

Concerning computational cost, because Walsh transform for each y_i is performed off-line, it does not affect search efficiency. However, Walsh transform for input x must be performed on-line so that it would affect search efficiency to some extent. By using fast Walsh transform [3], it only needs $k\times\log_2(k)$ additions (±) for transforming input x compared to $k\times(k-1)$ additions (±) by its definition. For most common 4×4 block size, to completely transform x into Walsh domain needs $16\times\log_2(16)=64$ additions (±) while to transform x to just get the first three elements as suggested in [9] needs $3\times(16-1)=45$ additions (±). It is not a great difference. Of course, only *once* on-line Walsh transform for x is sufficient.

On the other hand, if a rejection test by Eq.7 is successful for $j\le 3$, there would be no difference in computational cost. However, when all rejection tests by Eq.7 fail for j=1,2,3, unlike [9], this paper completely reuses the obtained distortion $(z_1-w_{i,1})^2+(z_2-w_{i,2})^2+(z_3-w_{i,3})^2$, which means to compute $d^2(z, w_i)$ from the 4th element by PDS method because all computations are conducted in Walsh domain now. This is an obvious saving for computational cost because it is profitable to all candidate codeword w_i.

Furthermore, because the absolute value operation (in practice, comparison operation is used) in Eq.4 is much lighter than the multiplication operation in Eq.7, it would be beneficial to use Eq.4 before Eq.7 for rejection tests. However, Eq.4 also checks $|z_j-w_{i,j}|\ge d_{\min}\quad j=1,2\ or\ 3$ one by one so that it is rather inefficient. In mathematics, for any m-dimensional vector $a=[a_1, a_2, \dots, a_m]^T$, the inequality $\sqrt{m}\times\|a\|_2\ge\|a\|_1$ holds, where $\|\cdot\|_1$ means the L_1 norm, $\|\cdot\|_2$ means the L_2 norm of a vector. Based on this inequality together with Eq.4 and Eq.7, it is easy to derive

$$\begin{aligned}&|z_1-w_{i,1}|\ge d_{\min}\\&(|z_1-w_{i,1}|)+|z_2-w_{i,2}|\ge\sqrt{2}\times d_{\min}\\&(|z_1-w_{i,1}|+|z_2-w_{i,2}|)+|z_3-w_{i,3}|\ge\sqrt{3}\times d_{\min}\end{aligned}\tag{8}$$

Clearly, Eq.8 deals with the first three basis vectors of Walsh transform *together*. Because the value in the bracket is already computed, it only needs two "±" and two "cmp" operations for a successive new rejection test. Eq.8 is more efficient than Eq.4.

If all rejection tests fail after using Eq.8 firstly and Eq.7 secondly, real Euclidean distance can be computed in Walsh domain as

$$\begin{aligned}d^2(z,w_i)&=\left[(z_1-w_{i,1})^2+(z_2-w_{i,2})^2+(z_3-w_{i,3})^2\right]+\sum_{j=4}^{k}(z_j-w_{i,j})^2\\&=d^2(x,y_i)\end{aligned}\tag{9}$$

In Eq.9, because the value in the square bracket is known after using Eq.7, it is definitely lighter to compute $d^2(z, w_i)$ in Walsh domain than to compute $d^2(x, y_i)$ in spatial domain.

In summary, the winner search is proposed as three stages in this paper. The first stage is to use Eq.8 for realizing three rejection tests in an incremental way. The second stage is to use Eq.7 for realizing two rejection tests in an incremental way because $(z_1-w_1)^2\ge d_{\min}^2$ and $|z_1-w_1|\ge d_{\min}$ are the same powerful. If both of them fail, the third stage is to use Eq.9 to compute real Euclidean distance for a final rejection test as $d^2(z,w_i)\ge d_{\min}^2$. No extra memories are needed.

Based on the discussions above, a search flow can be summarized as follows:

(1) Transform each y_i into Walsh domain off-line to get w_i by fast Walsh transform algorithm;

(2) Sort all codewords w_i by the first element $w_{i,1}$ in ascending order off-line and then rearrange all w_i along this order;

(3) For an input x, transform it into Walsh domain to get z by fast Walsh transform algorithm on-line as well;

(4) Find an initial nearest (NN) codeword w_N among sorted codewords $\{w_i|i=1,2, \dots, N_c\}$ by using a binary search, which is the closest codeword in terms of the difference between the first elements as $|z_1-w_{i,1}|\Rightarrow\min$. It needs $\log_2(N_c)$ times comparisons. Then compute "so far" $d_{\min}^2=d^2(z,w_N)$, $d_{\min}=\sqrt{d_{\min}^2}$, $\sqrt{2}d_{\min}$, $\sqrt{3}d_{\min}$ and temporally store them for future comparisons in each rejection test. This step needs (2k−1) additions (±), (k+2) multiplications (×) and one square root operations;

(5.1) Continue the winner search up and down around w_N one by one. Once $|z_1-w_{i,1}|\ge d_{\min}$ holds, terminate search for the upper part of sorted codebook $\{w_i|i=1,2, \dots, N_c\}$ 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 $(|z_1-w_{i,1}|)+|z_2-w_{i,2}|\ge\sqrt{2}\times d_{\min}$, $(|z_1-w_{i,1}|+|z_2-w_{i,2}|)+|z_3-w_{i,3}|\ge\sqrt{3}\times d_{\min}$, $(z_1-w_{i,1})^2+(z_2-w_{i,2})^2\ge d_{\min}^2$ $(z_1-w_{i,1})^2+(z_2-w_{i,2})^2+(z_3-w_{i,3})^2\ge d_{\min}^2$ and $d^2(z,w_i)\ge d_{\min}^2$ is true or not step by step. If any test is true, reject y_i safely;

(6) If all tests fail for a rejection, it implies that current w_i is a better-matched codeword, then update $d^2_{\min}$ by $d^2(z, w_i)$ and all $d_{\min}$, $\sqrt{2}d_{\min}$, $\sqrt{3}d_{\min}$ again. Meanwhile, update the corresponding winner index "so far" by current index "i". Then, return to Step (5.1) to test next codeword.

5. FURTHER DISCUSSION

In order to avoid using inverse Walsh transform at the decoder, it is preferred to store the codebook in spatial domain rather than in Walsh domain at the receiver. Because the codebook in Walsh domain at the transmitter that is sorted by $w_{i,1}$ in ascending order has a *one-to-one* mapping relation with the codebook in spatial domain at the receiver that is sorted by the mean in ascending order (Note: $w_{i,1}=\sqrt{k}\times My_i$), it is better to store the codebook in spatial domain that is sorted by the mean My_i in ascending order at the receiver. It is guaranteed that the winner index found in Walsh domain at the transmitter points to the same codeword at the receiver so that this index can be sent directly.

In addition to Eq.7, the previous work [9] also proposed a rejection rule in Walsh domain as

$$(z_1-w_{i,1})^2+(V_x-V_{y_i})^2\ge d_{\min}^2\tag{10}$$

where $V_x=d(x,z_1W_{k,1}^T)$, $V_{y_i}=d(y_i,w_{i,1}W_{k,1}^T)$. Because $z_1=\sqrt{k}Mx$, $w_{i,1}=\sqrt{k}My_i$, $z_1W_{k,1}^T=[Mx,Mx,\cdots,Mx]^T$, $w_{i,1}W_{k,1}^T=[My_i,My_i,\cdots,My_i]^T$ hold according to the definition of Walsh transform, Eq.10 is actually equivalent to Eq.6. They are the same powerful for a

rejection test. In other words, there is not any advantage to use both the projected component and its orthogonal component in Walsh domain. Only the projected component is sufficient for exploiting the energy compaction property of Walsh transform. This paper will not take the orthogonal component into account.

6. EXPERIMENTAL RESULTS

To compare the search performance with previous work [6] (i.e. Eq.4) directly combined with [9] (i.e. Eq.7), which are most powerful by using partial Walsh transform, simulation experiments by using Eq.8, Eq.7 and Eq.9 one after another are conducted. Codebooks of size 256, 512 and 1024 are generated using 512×512, 8-bit Lena image as a training set [10]. Block size is 4×4.

TABLE 1
COMPARISON OF TOTAL COMPUTATIONAL COST PER INPUT VECTOR

Size	Method	Operation	Lena	F-16	Pepper	Baboon
256	Full search	±	7936	7936	7936	7936
		×	4096	4096	4096	4096
		cmp	256	256	256	256
		sqrt	0	0	0	0
	Previous work [6]+[9]	±	262.2	212.8	285.2	743.0
		×	123.3	98.7	133.1	342.1
		cmp	132.0	109.7	147.3	407.7
		sqrt	2.9	2.2	2.8	3.1
	This work	±	217.3	177.0	235.4	590.8
		×	86.1	69.2	91.9	215.9
		cmp	128.3	104.5	141.9	385.4
		sqrt	2.9	2.2	2.8	3.1
512	Full search	±	15872	15872	15872	15872
		×	8192	8192	8192	8192
		cmp	512	512	512	512
		sqrt	0	0	0	0
	Previous work [6]+[9]	±	396.3	350.5	471.7	1389.4
		×	182.9	159.3	216.5	634.7
		cmp	213.6	195.3	259.2	783.0
		sqrt	3.2	2.6	3.0	3.6
	This work	±	323.1	286.2	382.4	1094.3
		×	121.8	105.9	142.3	388.9
		cmp	206.6	184.9	248.5	738.1
		sqrt	3.2	2.6	3.0	3.6
1024	Full search	±	31744	31744	31744	31744
		×	16384	16384	16384	16384
		cmp	1024	1024	1024	1024
		sqrt	0	0	0	0
	Previous work [6]+[9]	±	556.7	564.7	761.0	2395.3
		×	250.7	251.2	342.1	1078.5
		cmp	321.7	335.4	443.9	1399.7
		sqrt	3.6	3.0	3.5	4.0
	This work	±	451.8	456.6	611.8	1875.6
		×	161.6	160.2	216.6	642.0
		cmp	311.7	317.8	425.3	1315.6
		sqrt	3.6	3.0	3.5	4.0

Search efficiency is evaluated by total computational burden in terms of the number of addition (±), multiplication (×), comparison (cmp) and square root (sqrt) operations per input vector with FS as a relative baseline, which consists of (1) on-line Walsh transform for input x; (2) finding the initial best-matched codeword w_N and computing the initial d^2_{min}, d_{min}, $\sqrt{2}\times d_{min}$, $\sqrt{3}\times d_{min}$; (3) computing all test conditions for a possible rejection based on Eq.8, Eq,7 and Eq.9 according to the search flow suggested in Section 4; (4) updating "so far" d^2_{min}, d_{min}, $\sqrt{2}\times d_{min}$, $\sqrt{3}\times d_{min}$ again if current w_i is a better-matched codeword.

From Table 1, it can be seen that the total computational cost can be reduced compared to a direct combination of Eq.4 and Eq.7 proposed in [6] and [9], especially multiplication (×) operations can be reduced obviously.

7. CONCLUSION

In this paper, a fast search method that is conducted only in Walsh domain is proposed. Two advantages are achieved. First, extra memory requirement is completely avoided. As a result, k memories are sufficient in total for representing a k-dimensional vector. Second, total computational cost is reduced by enhancing the rejection power of DHSS method (i.e. Eq.8) and by completely reusing the distortion obtained from the first three dimensions in Walsh domain (i.e. Eq.9). The proposed method is very promising and practical to fast VQ encoding.

8. 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] R.Costantini, J.Bracamonte, G.Ramponi, J.L. Nagel, M.Ansorge and F.Pellandini, "A low-complexity video coder based on the discrete Walsh-Hadamard transform," *Proc. European Signal Processing Conf. (EUSIPCO)*, pp.5-8, 2000.

[3] K.G. Beauchamp, *Application of Walsh and related functions*, Academic Press London, 1984.

[4] 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.

[5] C.D.Bei and R.M.Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," *IEEE Trans. Commun*., vol.33, pp.1132-1133, Oct. 1985.

[6] S.C.Tai, C.C.Lai and Y.C.Lin, "Two fast nearest neighbor searching algorithms for image vector quantization," *IEEE Trans. Commun.,* vol. 44, pp.1623-1928, Dec. 1996.

[7] S.Baek, M.Bae and K.Sung, "A fast vector quantization encoding algorithm using multiple projection axes," *Signal Processing*, pp.89-92, 1999.

[8] 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.

[9] S.Baek and K.Sung, "Two fast nearest neighbor searching algorithms for vector quantization," *IEICE Trans. Fundamentals,* vol.84-A, pp.2569-2575, 2001.

[10] 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.