An Improved Full-Search-Equivalent Vector Quantization Method Using the Law of Cosines

著者	小谷　光司
journal or publication title	IEEE signal processing letters
volume	11
number	2
page range	247-250
year	2004
URL	http://hdl.handle.net/10097/34915

An Improved Full-Search-Equivalent Vector Quantization Method Using the Law of Cosines

Zhibin Pan, *Member, IEEE*, Koji Kotani, *Member, IEEE*, and Tadahiro Ohmi, *Fellow, IEEE*

***Abstract*—Vector quantization (VQ) is a well-known signal compression method. In VQ, the search process to find the winner for an input vector either at the codebook generation stage or the VQ encoding stage is extremely time consuming. By using the law of cosines to estimate the Euclidean distance first, Mielikainen has developed a highly efficient full-search-equivalent algorithm. However, some computational redundancies still exist in it. In this letter, we introduce an additional new estimation for the Euclidean distance and then optimize the computing way given by Mielikainen. Mathematical analyses show that our proposed search method can improve Mielikainen's method. And experimental results of VQ encoding demonstrate that the proposed method is very search effective.**

***Index Terms*—Euclidean distance estimation, fast search, the law of cosines, vector quantization (VQ).**

I. Introduction

VECTOR QUANTIZATION (VQ) [1] is a classical but still very promising method for signal compression, especially for images such as computer graphics or digitized documents that have many abrupt discontinuities. In order to use VQ, a codebook is necessary from the very beginning, and it can usually be generated by the famous Linde–Buzo–Gray method [2]. Once the codebook is available, encoding by VQ is possible. The search process to find the closest item (winner) for a specific input vector via a large amount of Euclidean distance computation becomes the time bottleneck of VQ.

For an input vector $v = (v_1, v_2, \ldots, v_k)$ and a code vector $u = (u_1, u_2, \ldots, u_k)$, which is in the codebook C of size N_c, the difference between v and u can be measured by squared Euclidean distance as

$$d^2(u,v) = \sum_{i=1}^{k}(u_i - v_i)^2 = \|u - v\|^2. \tag{1}$$

(Note: $d^2(u,v)$ instead of $d(u,v)$ is used to represent the squared Euclidean distance for convenience.)

Therefore, the winner can be determined straightforwardly by

$$d^2(u_w, v) = \min_{u \in C}\left[d^2(u,v)\right] \tag{2}$$

where u_w is the winner. This is called a full search (FS) over the whole codebook. FS is extremely time consuming due to performing N_c times Euclidean distance computation. If part of the N_c times distance computation can be avoided by using some lighter computations instead but exactly the same winner can be guaranteed, the method is defined as FS-equivalent. Many FS-equivalent fast search methods have already been developed to determine the winner [3]–[6]. FS-equivalent methods can be applied to both codebook generation and VQ encoding in the same way. This letter will deal with VQ encoding only.

II. Previous Work

In [4], an FS-equivalent search method is proposed. By the law of cosines, $d^2(u,v)$ can be expressed as

$$d^2(u,v) = \|u - v\|^2 = \|u\|^2 + \|v\|^2 - 2\|u\|\|v\|\cos(\theta) \tag{3}$$

where θ is the angle between u and v. Since all elements of u or v are positive for image vectors, $0^\circ \le |\theta| \le 90^\circ$ holds. From (3), it is clear that the difference vector $(u - v)$ is coupled by θ that makes the offline computation for $\cos(\theta)$ impossible. To solve this problem, Mielikainen [4] has set up a fixed reference vector x so as to decouple the effect of θ. This purposely generated x is the key point in [4]. Because $u, v,$ and x intersect at the origin, and x is generally not within the plane determined by u and v, then a *trihedral angle* is formed by $u, v,$ and x in R^k space. By introducing x, two additional angles between the (u,x) pair and (v,x) pair can be obtained. Let θ_1 be the angle between (u,x) and θ_2 be the angle between (v,x). Mielikainen [4, pp. 175, line 19 on the right-hand side] shows an observation that says $\theta \ge |\theta_1 - \theta_2|$. Therefore, $\cos(\theta) \le \cos(|\theta_1 - \theta_2|)$ holds because the cosine is an even function and monotonically decreases over $[0^\circ, 90^\circ]$. Based on the discussions above, Mielikainen [4] proposed a very computationally light estimation of $d_*^2(u,v)$ as

$$d_*^2(u,v) = \|u\|^2 + \|v\|^2 - 2\|u\|\|v\|\cos(|\theta_1 - \theta_2|) \le d^2(u,v). \tag{4}$$

During the winner search process, suppose the achieved minimum distance "so far" is $d_{\min}^2(u,v)$. If $d_*^2(u,v) \ge d_{\min}^2(u,v)$, then $d^2(u,v) \ge d_{\min}^2(u,v)$ holds definitely. Thus, the current code vector u cannot be a winner and can be rejected safely. Equation (4) can be expanded into

$$d_*^2(u,v) = \|u\|^2 + \|v\|^2 - 2\|u\|\|v\| \cdot [\cos(\theta_1)\cos(\theta_2) + \sin(\theta_1)\sin(\theta_2)] \tag{5}$$

Manuscript received December 25, 2002; revised May 1, 2003. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Pamela C. Cosman.

Z. Pan and T. Ohmi are with the New Industry Creation Hatchery Center, Tohoku University, Sendai 980-8579, Japan (e-mail: pzb@fff.niche.tohoku.ac.jp).

K. Kotani is with the Department of Electronic Engineering, Graduate School of Engineering, Tohoku University, Sendai 980-8579, Japan.

Digital Object Identifier 10.1109/LSP.2003.821756

where

$$\cos(\theta_1) = \langle u, x\rangle / (\|u\|\|x\|)$$
$$\cos(\theta_2) = \langle v, x\rangle / (\|v\|\|x\|)$$

and $\langle\ \rangle$ means the inner product. According to (5), the coupling between (u, v) can be released by introducing (u, x) and (v, x) pairs and the offline computation becomes possible.

For a code vector, $\|u\|^2$, $2 \times \|u\|$, $\cos(\theta_1)$, and $\sin(\theta_1)$ can be computed offline, and they are saved using four extra memories. For the input vector in VQ encoding, $\|v\|^2$, $\|v\|$, $\cos(\theta_2)$, and $\sin(\theta_2)$ are computed online but just once. From (5), it is clear that $d_*^2(u, v)$ needs four multiplication $(\times)$ and three addition $(\pm)$ operations online. It is still more complex.

III. Improved Search Method

First, for a trihedral angle constituted by $u, v,$ and x, the *trihedral theorem* in solid geometry states that "of the three plane angles that form a trihedral angle, any two are together greater than the third." Therefore, the observation $\theta \geq |\theta_1 - \theta_2|$ that is the core of [4] can be proven mathematically. Because both $\theta + \theta_1 \geq \theta_2$ and $\theta + \theta_2 \geq \theta_1$ hold, $\theta \geq |\theta_1 - \theta_2|$ is identically correct.

Second, from (3), it is easy to get another estimation $d_\#(u, v)$ before $d_*^2(u, v)$ by letting $\cos(\theta) = 1$

$$d_\#^2(u, v) = \|u\|^2 + \|v\|^2 - 2\|u\|\|v\| \Rightarrow d_\#(u, v) = |\|u\| - \|v\||. \tag{6}$$

$d_\#(u, v)$ is proposed as an additional new estimation in this letter. It is much lighter computationally than $d_*^2(u, v)$. Combining (6) with (4), it is clear that

$$d(u, v) \geq d_*(u, v) \geq d_\#(u, v). \tag{7}$$

Equation (7) is the key to the fast search method in this letter. Both $d_{\min}(u, v)$ and $d_{\min}^2(u, v)$ are kept online. The search is executed as follows. 1) If $d_\#(u, v) \geq d_{\min}(u, v)$, which is a new check, then reject current code vector safely. 2) Else, check whether $d_*^2(u, v) \geq d_{\min}^2(u, v)$ holds, which is the same as [4]. If this inequality holds, then reject current code vector safely as well. 3) When both $d_\#(u, v)$ and $d_*^2(u, v)$ checks fail, compute squared Euclidean distance $d^2(u, v)$. If $d^2(u, v) \geq d_{\min}^2(u, v)$ holds, then reject the current code vector. Otherwise, update the winner index and $d_{\min}(u, v)$, $d_{\min}^2(u, v)$ accordingly.

$d_\#(u, v)$ is very light computationally. Then, the third problem is how to compute $d_*^2(u, v)$ more efficiently with less memory. Based on the concept of vector projection, for the (u, x) pair, we have $u_x = \|u\|\cos(\theta_1)$ and $u_{x,\perp} = \|u\|\sin(\theta_1)$, where u_x is the projection of u to x, and $u_{x,\perp}$ is the orthogonal component. For the (v, x) pair, $v_x = \|v\|\cos(\theta_2)$ and $v_{x,\perp} = \|v\|\sin(\theta_2)$ imply the same thing. Thus, (5) can be reformed as

$$d_*^2(u, v) = \|u\|^2 + \|v\|^2 - 2(u_x v_x + u_{x,\perp} v_{x,\perp}). \tag{8}$$

Furthermore, any vector w can be expanded with x as a reference vector in an orthogonal way as $\|w\|^2 = w_x^2 + w_{x,\perp}^2$. Therefore, (8) can be rewritten as

$$d_*^2(u, v) = \left(u_x^2 + u_{x,\perp}^2\right) + \left(v_x^2 + v_{x,\perp}^2\right) - 2(u_x v_x + u_{x,\perp} v_{x,\perp})$$

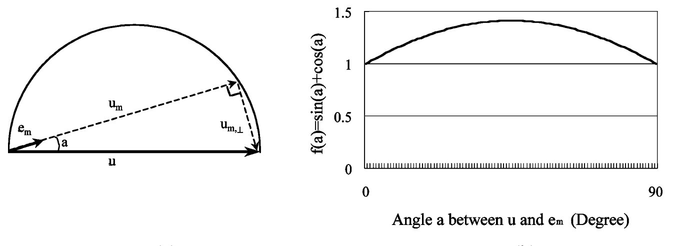

Fig. 1. (a) Concept of code vector decomposition. (b) How $f(a)$ varies with the angle a in a symmetric way.

$$= (u_x - v_x)^2 + (u_{x,\perp} - v_{x,\perp})^2. \tag{9}$$

Because x is only used for generating the two angles θ_1 and θ_2, any length for it is allowed. Without losing generality, x can be selected as the mth standard basis vector $e_m = (0, 0, \ldots\ldots, 1_{(m)}, \ldots\ldots, 0)$, $m \in [1, k]$. Then, (9) becomes

$$d_*^2(u, v) = (u_m - v_m)^2 + (u_{m,\perp} - v_{m,\perp})^2 \tag{10}$$

where u_m and v_m are the mth element of u and v (i.e., their projections to e_m), respectively. $u_{m,\perp} = \sqrt{(\|u\|^2 - u_m^2)}$ and $v_{m,\perp} = \sqrt{(\|v\|^2 - v_m^2)}$ are their orthogonal components. In this way, only $u_{m,\perp}$ has to be computed offline and saved for the code vector u. $d_*^2(u, v)$ just needs two multiplication $(\times)$, three addition $(\pm)$ operations and one extra memory. In addition, (10) avoids using the cosine and sine functions that will affect computation precision near zero very much. Equation (10) is computationally lighter (about half) and more memory efficient (1/4) than (5) as suggested in [4].

Finally, the fourth problem is how to choose e_m optimally from the set of standard basis vectors $\{e_j : j = 1 \sim k\}$. The object is to let the value of $d_*^2(u, v)$ become as large as possible so as to reject a candidate code vector more easily. Returning to (8) for $x = e_m$, only $(u_m \times v_m + u_{m,\perp} \times v_{m,\perp})$ is changeable regarding m. If $(u_m \times v_m + u_{m,\perp} \times v_{m,\perp})$ is smaller, $d_*^2(u, v)$ will be larger. Because u and v are independent and randomly distributed in R^k space, it is very difficult to deal with them simultaneously to reach a global optimization. In this letter, we just take u_m and $u_{m,\perp}$ components of the known code vector u into account while treating v_m and $v_{m,\perp}$ as their random weights for simplicity, i.e., let $(u_m + u_{m,\perp})$ be smaller so that on average $(v_m \times u_m + v_{m,\perp} \times u_{m,\perp})$ could also be smaller statistically.

Suppose a is the angle between e_m and u. Because both of them are positive vectors, $|a| \in [0°, 90°]$ holds. Then $(u_m + u_{m,\perp})$ normalized by $\|u\|$ is

$$f(a) = \frac{(u_m + u_{m,\perp})}{\|u\|} = \cos(a) + \sin(a). \tag{11}$$

How u is decomposed into u_m, $u_{m,\perp}$ is demonstrated in Fig. 1(a). The function $f(a)$ that is mirrored with the $a = 45°$ line is plotted in Fig. 1(b).

From Fig. 1(b), it is clear that when $a = 0°$ or $a = 90°$, $(u_m + u_{m,\perp}) = \|u\|$ reaches min, and when $a = 45°$, $(u_m + u_{m,\perp}) = \sqrt{2} \times \|u\|$ reaches max. Therefore, in order to let $(u_m + u_{m,\perp}) \Rightarrow \min$, it is necessary to let $a \Rightarrow 0°$ or $a \Rightarrow 90°$ that are symmetric mathematically. Because $a \Rightarrow 0°$ means

$u_m \Rightarrow \|u\|$ or equivalently $u_j \Rightarrow 0$ for $j = 1 \sim k$ but $j \neq m$, it is a very unlikely event in practice. While for $a \Rightarrow 90^\circ$, it just implies to let $u_m \Rightarrow 0$ but does not concern other u_j, it is usually a higher possibility event. As a result, the criteria for choosing m in this letter is proposed as

$$u_m = \min_{j\in[1,k]}(u_j). \tag{12}$$

Thus, one more extra memory is needed to keep selected m for each code vector. However, for each selected standard basis vector e_m, $v_{m,\perp}$ has to be computed once online. The square root computation is time consuming. In practice, there exists a tradeoff between how many various reference vectors to choose totally and how much computational burden can be reduced accordingly. This tradeoff depends on how large the ratio is for the cost of one square root computation and the cost of one Euclidean distance computation. Generally speaking, the number of reference vectors could be larger when the dimension of a vector is high. In practice, a threshold value can be set up for $u_m/\|u\|$ by experiment, but it strongly relies on the codebook itself. If the threshold value is not sufficiently small, a fixed e_m (e.g., e_1) is used instead to avoid the corresponding computation for $v_{m,\perp}$.

IV. Further Discussion

By using the projection axis, a very fast encoding method for VQ has been proposed in [5]. The method in [5] originates from the concept of the central axis in R^k space, the mean, and the variance of a vector. According to [5], the central axis is defined as follows: if any point $p = (p_1, p_2, \ldots, p_k)$ on a line L satisfies $p_1 = p_2 = \ldots = p_k$, then L is called the central axis of R^k space. For a vector $w = (w_1, w_2, \ldots, w_k)$, its projection onto L will be the point $L_w = (\mu_w, \mu_w, \ldots, \mu_w)$, where μ_w is the mean of w. Then, three points w, L_w, and the origin $O = (0, 0, \ldots, 0)$ constitute a triangle in R^k space. Clearly, the projection of w onto $L(L_w O)$ is $d(L_w, O) = \sqrt{k} \times \mu_w$, and its orthogonal component (wL_w) is $d(w, L_w) = \sqrt{\sum_{i=1}^{k}(w_i - \mu_w)^2} \overset{\text{def}}{=} V_w$. By using the information of the mean and the variance together, Baek and Jeon [5] developed a very powerful inequality

$$d^2(u,v) \geq k(\mu_u - \mu_v)^2 + (V_u - V_v)^2. \tag{13}$$

The right-hand term of (13) is computationally light. If it is larger than $d^2_{\min}(u,v)$, the code vector u can be rejected safely. On the other hand, if we use the central axis L as the reference vector x, based on the discussions above, it is clear that (9) becomes

$$\begin{aligned} d^2_*(u,v) &= (\sqrt{k}\mu_u - \sqrt{k}\mu_v)^2 + (V_u - V_v)^2 \\ &= k(\mu_u - \mu_v)^2 + (V_u - V_v)^2. \end{aligned} \tag{14}$$

Therefore, it reaches the conclusion that (9) based on the law of cosines is the general version of (13) proposed in [5] for $x = L$. In summary, the method using the law of cosines can cover all fast VQ encoding methods that are based on the concept of using a projection axis. Thus, the only problem is how to choose a reference vector or a projection axis optimally. Apparently, using e_m as the reference vector x can save one memory for u_x and online projection operation for v_x. It is the simplest way for choosing x. In contrast, the central axis proposed in [5] mainly exploits the potential power of using the statistical averaging. Baek and Jeon [5] also concluded that (13) is advantageous in the case of the input vectors concentrating on the central axis, which implies that the image is very low-detailed. Otherwise, an appropriate new projection axis instead of the central axis has to be found by another method.

Mathematically, if the reference vector x is within the plane determined by the input vector v and a code vector u, then $\theta = |\theta_1 - \theta_2|$ holds according to the trihedral theorem. Then, $d_*(u,v) = d(u,v)$ is true as well. Therefore, this x must be the optimal reference vector for u, but it is not the sole one. Unfortunately, it is impractical to use online processing to find an optimal reference vector x for each code vector. And the detailed discussion on how to suitably choose a suboptimal x that can approach the optimal one as a limit at a smaller cost is out of the scope of this letter. As a simplified case, however, a possible suboptimal choosing way for x is given in (12).

TABLE I
Ratio of Remaining Code Vectors on Average After the $d_{\#}(u,v)$ Check Compared to FS (100%)

Test image	Lena	Pepper	F-16	Baboon
R-$d_{\#}$(%)	5.316	5.651	6.446	15.497

V. Experimental Results

In this letter, VQ encoding is implemented to verify the search efficiency of the improved method. The method can be straightforwardly applied to codebook generation. Because [4] has not shown how x is selected, it is impractical to make a direct comparison between these two methods. However, it is obvious that the improved method mathematically outperforms the method of [4] in principle when computational burden and memory are taken into consideration.

Simulation experiments with MATLAB are executed on a personal computer. Four typical eight-bit, 512 $\times$ 512 standard images (Lena, F-16, Pepper, and Baboon) that feature very different details are used to test the search efficiency of the improved method. Block size is 4 $\times$ 4. A codebook of size 1024 is used that has been developed by Kohonen's self-organizing map (SOM) combined with perceptual property of human vision system in [7]. The search efficiency is evaluated by how many code vectors are left after rejection check using $d_{\#}(u,v)$ and the succeeding $d^2_*(u,v)$ for encoding an image. The comparison takes FS (100%) as a relative base line.

From Table I, it is clear that the $d_{\#}(u,v)$ check is most efficient for rejecting. Depending on image details, it can reject about 85% or more code vectors by a very light computation.

If the $d_{\#}(u,v)$ check fails, then the $d^2_*(u,v)$ check is used. According to the discussions for (11), four typical ways of selecting e_m are tested:

$$u_m = \min_{j\in[1,k]}(u_j) \qquad u_m = \max_{j\in[1,k]}(u_j)$$
$$u_m = \operatorname*{median}_{j\in[1,k]}(u_j) \qquad u_m = u_1.$$

TABLE II
RATIO OF REMAINING CODE VECTORS ON AVERAGE AFTER THE $d_*^2(u,v)$ CHECK DEPENDING ON e_m SELECTION COMPARED TO FS (100%)

Test image	Lena	Pepper	F-16	Baboon
R-$d^2_{*\mathrm{min}}$ (%)	3.715	4.045	4.585	12.50
R-$d^2_{*\mathrm{max}}$ (%)	3.880	4.196	5.054	13.35
R-$d^2_{*\mathrm{median}}$ (%)	4.523	4.868	5.485	14.55
R-$d^2_{*\mathrm{dim1}}$ (%)	4.041	4.374	5.075	13.55

TABLE III
RATIO OF REMAINING CODE VECTORS ON AVERAGE AFTER THE $d_*^2(u,v)$ CHECK COMPARED TO THOSE AFTER THE $d_\#(u,v)$ CHECK

Test image	Lena	Pepper	F-16	Baboon
R-$d^2_{*\mathrm{min}}$/ R-$d_\#$ (%)	69.88	71.58	71.12	80.66

The threshold value constraint for $u_m/\|u\|$ is not incorporated here. The effect of e_m selection is summarized in Table II.

From Table II, it is true that for search efficiency the best way is to select m using min and the worst way is to select m using median. Equation (12) is a practical choice for m.

Therefore, the fixed reference vector x (i.e., e_m) could be suboptimally selected according to (12) for its simplicity. Under this condition, the power of $d_*^2(u,v)$ check can be evaluated by the ratio of $R - d^2{}_{*\,\mathrm{min}}/R - d_\#(\%)$. A smaller ratio is better.

From Table III, it can be seen that only about 20% to 30% candidate code vectors that could not be rejected by $d_\#(u,v)$ are successfully rejected by $d_*^2(u,v)$. This is because the remaining candidate code vectors after $d_\#(u,v)$ check are already very close to the input vector and the rejection becomes very difficult. Even though what $d_\#(u,v)$ can reject can also be rejected by $d_*^2(u,v)$, considering the online computational burden, $d_\#(u,v)$ is good to check before $d_*^2(u,v)$.

VI. CONCLUSION

In this letter, a mathematical explanation to the observation of $\theta \geq |\theta_1 - \theta_2|$ using the trihedral theorem is shown. Before the $d_*^2(u,v)$ check, an additional new estimation $d_\#(u,v)$ is suggested that is very light computationally and most search efficient. Concerning about how to compute $d_*^2(u,v)$ efficiently, a suboptimal selection for the fixed reference vector x is proposed as the mth ($m \in [1,k]$) standard basis vector e_m, where m is determined by

$$u_m = \min_{j\in[1,k]}(u_j).$$

The experimental results of VQ encoding show that the improved method is very search effective. Furthermore, the proposed method can also be viewed as a general version for any fast search VQ method based on the concept of projection axis. And the mathematical criteria for choosing the optimal rather than suboptimal reference vector x is to let it lie within the plane determined by u and v. This method can be applied to codebook generation directly.

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] Y. Linde, A. Buzo, and R. M. Gray, “An algorithm for vector quantizer design,” *IEEE Trans. Commun.*, vol. 28, pp. 84–95, Jan. 1980.

[3] 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, pp. 413–416, July 1992.

[4] J. Mielikainen, “A novel full-search vector quantization algorithm based on the law of cosines,” *IEEE Signal Processing Lett.*, vol. 9, pp. 175–176, June 2002.

[5] S. J. Baek, B. K. Jeon, and K. M. Sung, “A fast encoding algorithm for vector quantization,” *IEEE Signal Processing Lett.*, vol. 4, pp. 325–327, Dec. 1997.

[6] Z. Pan, K. Kotani, and T. Ohmi, “A hierarchical fast encoding algorithm for vector quantization with PSNR equivalent to full search,” in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. I, 2002, pp. 797–800.

[7] 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.