Fast VQ Encoding by an Efficient Kick-Out Condition

Kuang-Shyr Wu and Ja-Chen Lin

***Abstract*—A new fast approach to the nearest codeword search using a single kick-out condition is proposed. The nearest codeword found by the proposed approach is identical to the one found by the full search, although the processing time is much shorter. The principle is to bypass those codewords which satisfy the proposed kick-out condition without the actual (and time-consuming) computation of the distortions from the bypassed codewords to the query vector. Due to the efficiency and simplicity of the proposed condition, a considerable saving of the central processing unit time needed to encode a data set (using a given codebook) can be achieved. Moreover, the memory requirement is low. Comparisons with some recent works are included to show these two benefits.**

***Index Terms*—Fast searching, nearest codeword.**

I. Introduction

Signal compression using vector quantization (VQ) has been widely applied to speech and image coding [1], [2]. The source signal is typically segmented into individual vectors, and then, for each vector of the source signal, the VQ encoder searches through a predesigned codebook $\mathbf{Y} = \{\mathbf{y_i}\}_{i=1}^{N}$ to find the nearest codeword to represent that vector. Compression is achieved by sending the indices of the nearest codewords to the receiver. The (lossy) reconstruction of the source signal is then established at the decoder end by applying a table look-up technique. Compared with other compression techniques, VQ has a very simple decoder which is quite suitable for broadcasting video services [3]. However, the searching speed of the nearest codeword often limits the applications of VQ in the real-time compression system. The situation becomes severer when the vector dimension k or codebook size N increases [4]. Therefore, accelerating the searching speed of the nearest codeword has become an attractive topic in VQ study.

To reduce the computation burden, some authors provided certain kick-out conditions to speed up the searching process, while keeping the reconstructed signals the same as that of the full search [5]–[10]. Each of these kick-out conditions can accelerate the codebook generation [11], [12] or encoding process [13]–[15]. To simplify the discussion, we only focus on the acceleration of the encoding process and assume that the codebook is already given. The technique introduced in this study, however, can be applied directly to accelerate the Linde, Buzo, and Gray (LBG) codebook generation [16] process, because in each of the LBG iteration, one of the two main steps is to find the nearest codeword (using the temporary codebook) for each data vector.

Manuscript received January 5, 1996; revised June 17, 1999. This work was supported by the National Science Council, R.O.C., under Contract NSC 88-2213-E009-067. This paper was recommended by Associate Editor C.-H. Wei.

The authors are with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan 30010, R.O.C.

Publisher Item Identifier S 1051-8215(00)01634-7.

II. Some Existing Techniques

To reduce the time complexity of VQ encoding, the form "if the kick-out condition is satisfied, then kick out the codeword being discussed" is often used by researchers to bypass (kick-out) many codewords without computing the distortion from those codewords to the query vector. We briefly review below some previous works of this kind.

In the k-dimensional vector space, let a query vector $\mathbf{x} = (x_1, x_2, \cdots, x_k)$ and a codebook $\mathbf{Y} = \{\mathbf{y_i}\}_{i=1}^{N} = \{(y_{i1}, y_{i2}, \cdots, y_{ik})\}_{i=1}^{N}$ of size N be given. When the squared Euclidean distance is used as the distortion measure, the distortion between $\mathbf{x}$ and a codeword $\mathbf{y_i} \in \mathbf{Y}$ is

$$d(\mathbf{x}, \mathbf{y_i}) = \sum_{j=1}^{k} (x_j - y_{ij})^2 = \|\mathbf{x} - \mathbf{y_i}\|^2. \tag{1}$$

The goal of the nearest codeword search process is to find a $\mathbf{y} \in \mathbf{Y}$, such that

$$d(\mathbf{x}, \mathbf{y}) = \min_{i=1,\cdots,N} d(\mathbf{x}, \mathbf{y_i}).$$

Without loss of generality, assume that a part of the codebook has been inspected, and the "so far" smallest distortion is

$$d_{\min} = \min\{d(\mathbf{x}, \mathbf{y_l}) \mid \mathbf{y_l} \text{ has been inspected}\}.$$

Also, let the "so far" nearest codeword $\mathbf{y}_{\min} \in \{\mathbf{y_l} \mid \mathbf{y_l}$ has been inspected$\}$ be such that

$$d(\mathbf{x}, \mathbf{y}_{\min}) = d_{\min}. \tag{2}$$

We then want to know whether $d(\mathbf{x}, \mathbf{y_i})$ is smaller than $d_{\min}$, if $\mathbf{y_i}$ is not yet inspected.

As mentioned above, researchers often tried to find a sufficient condition, the so-called "kick-out condition," which, if satisfied, guarantees that $d(\mathbf{x}, \mathbf{y_i}) \geq d_{\min}$, and hence, rules out the possibility that codeword $\mathbf{y_i}$ can replace the "so far" nearest codeword $\mathbf{y}_{\min}$. The computation of $d(\mathbf{x}, \mathbf{y_i})$ is therefore bypassed if the sufficient condition is satisfied. Of course, the more conditions are used, the more likely $\mathbf{y_i}$ will be kicked out (and the higher overhead is loaded). One of the earliest approaches is known as the Partial Distance Elimination (PDE) [6], in which the premature-exit condition

$$\sum_{j=1}^{q} (x_j - y_{ij})^2 \geq d_{\min}, \quad \text{for some } q \leq k \tag{3}$$

was developed and used as the kick-out condition by Bei and Gray. A similar condition used by Soleymani and Morgera [7] to kick out $\mathbf{y_i}$ was

$$|x_j - y_{ij}| \geq \sqrt{d_{\min}}, \quad \text{for some } j \in \{1, \cdots, k\}. \tag{4}$$

Besides, many literature [4], [12], [14], [15] employed the Triangular Iinequality Elimination (TIE) approach, in which $\mathbf{y_i}$ is kicked out if it is too far away from the $\mathbf{y}_{\min}$ defined in (2), i.e., if

$$d(\mathbf{y}_{\min}, \mathbf{y_i}) \geq 4d_{\min}. \tag{5}$$

TIE is efficient for codeword search; however, it requires $O(N^2)$ memory space for the storage of all distortion between every pair of the codewords. In this study, the TIE is referred to as algorithm A1.

Torres and Huguet also developed two useful inequalities [9], i.e., kick out $\mathbf{y_i}$ if

$$||\mathbf{x}||^2 + ||\mathbf{y_i}||^2 - 2x_{\max}\sum_{j=1}^{k} y_{ij} \geq d_{\min} \tag{6}$$

or

$$||\mathbf{x}||^2 + ||\mathbf{y_i}||^2 - 2y_{i\max}\sum_{j=1}^{k} x_j \geq d_{\min} \tag{7}$$

is satisfied. Here, $x_{\max}$ is the maximum component of $\mathbf{x}$, i.e., $x_{\max} = \max\{x_1, x_2, \cdots, x_k\}$, and $y_{i\max}$ is the maximum component of $\mathbf{y_i}$, i.e., $y_{i\max} = \max\{y_{i1}, y_{i2}, \cdots, y_{ik}\}$. In fact, they combined these two inequalities with condition (3) (the PDE) to save computation time. This elegant method needs a reasonable $O(N)$ memory space; the only inconvenience is that the method requires a translation of the coordinate system so that all vectors being discussed have non-negative components. This algorithm will be referred to hereafter as A2.

Recently, Lin and Tai proposed a new approach called the integral projection method [5], [10] to accelerate the codeword search process when each vector is in fact an image block. They first define three kinds of integral projection for each $m \times m$ block $\mathbf{x} = \{x_{jl} \mid j = 1, \cdots, m; l = 1, \cdots, m\}$, which is a k-dimensional vector with $k = m^2$, as follows:

massive projection

$$mp = \sum_{j=1}^{m}\sum_{l=1}^{m} x_{jl} \tag{8}$$

vertical projections

$$vp_l = \sum_{j=1}^{m} x_{jl}, \quad \text{where } 1 \leq l \leq m \tag{9}$$

horizontal projections

$$hp_j = \sum_{l=1}^{m} x_{jl}, \quad \text{where } 1 \leq j \leq m. \tag{10}$$

Then, kick out the codeword $\mathbf{y_i}$ no matter which of the following three inequalities is satisfied:

$$[mp(\mathbf{x}) - mp(\mathbf{y_i})]^2 \geq m^2 d_{\min} \tag{11}$$

$$\sum_{l=1}^{m}[vp_l(\mathbf{x}) - vp_l(\mathbf{y_i})]^2 \geq m d_{\min} \tag{12}$$

$$\sum_{j=1}^{m}[hp_j(\mathbf{x}) - hp_j(\mathbf{y_i})]^2 \geq m d_{\min}. \tag{13}$$

The integral projection algorithm is referred to hereafter as A3.

III. Proposed Algorithm

The proposed kick-out condition and the related algorithm are presented in this section. First, (1) can be rewritten as

$$d(\mathbf{x}, \mathbf{y_i}) = \sum_{j=1}^{k} x_j^2 + \sum_{j=1}^{k} y_{ij}^2 - 2\sum_{j=1}^{k} x_j y_{ij} \tag{14}$$

$$= ||\mathbf{x}||^2 + ||\mathbf{y_i}||^2 - 2\sum_{j=1}^{k} x_j y_{ij}. \tag{15}$$

Although our goal is to find a codeword $\mathbf{y_i}$, $1 \leq i \leq N$, that minimizes (15), an equivalent (but time saving) goal is to find a codeword $\mathbf{y_i}$, $1 \leq i \leq N$, that minimizes

$$\begin{aligned} d_1(\mathbf{x}, \mathbf{y_i}) &= d(\mathbf{x}, \mathbf{y_i}) - ||\mathbf{x}||^2 \\ &= ||\mathbf{y_i}||^2 - 2\sum_{j=1}^{k} x_j y_{ij} \end{aligned} \tag{16}$$

because $||\mathbf{x}||^2$ is a common term in the distortion measure (15) from $\mathbf{x}$ to every codeword. In fact, evaluating (16) is also faster than evaluating (1) if we notice that $\{||\mathbf{y_i}||\}_{i=1}^{N}$ are known values stored in the preprocessing procedure.

For the remainder of this section, we will forget the squared Euclidean distortion system (1), i.e., (15), and concentrate on the minimization of the d_1 distortion system defined in (16). Again, assume that a part of codewords have been inspected, and

$$d_{1\min} = d_1(\mathbf{x}, \mathbf{y}_{\min}) = \min\{d_1(\mathbf{x}, \mathbf{y_l}) \mid \mathbf{y_l} \text{ has been inspected}\} \tag{17}$$

is the "so far" smallest d_1-distortion (cf. (2)). By (15) and (16), we also know that the "so far" nearest codeword $\mathbf{y}_{\min}$ minimizes not only $\{d_1(\mathbf{x}, \mathbf{y_l}) \mid \mathbf{y_l} \text{ has been inspected}\}$ but also $\{d(\mathbf{x}, \mathbf{y_l}) \mid \mathbf{y_l} \text{ has been inspected}\}$.

Before we go ahead, note that

$$d_1(\mathbf{x}, \mathbf{y_i}) \geq ||\mathbf{y_i}||^2 - 2||\mathbf{y_i}|| \cdot ||\mathbf{x}|| = ||\mathbf{y_i}||(||\mathbf{y_i}|| - 2||\mathbf{x}||) \tag{18}$$

is always true due to the Cauchy-Schwarz inequality. As a result, if a codeword $\mathbf{y_i}$ satisfies

$$||\mathbf{y_i}||(||\mathbf{y_i}|| - 2||\mathbf{x}||) \geq d_{1\min} \tag{19}$$

then $d_1(\mathbf{x}, \mathbf{y_i}) \geq d_{1\min}$ is guaranteed, and hence, $\mathbf{y_i}$ should be kicked out because $\mathbf{y_i}$ cannot be closer to $\mathbf{x}$ then $\mathbf{y}_{\min}$ [the "so far" nearest codeword defined in (17)] is.

For the query vector $\mathbf{x}$, the computation of $||\mathbf{y_i}||(||\mathbf{y_i}||-2||\mathbf{x}||)$ is quite simple, because the computations of $\{||\mathbf{y_i}||\}_{i=1}^{N}$ can be done in advance in the preprocessing procedure. For the reader's benefit, we give below the algorithm that utilizes the proposed kick-out condition to accelerate the encoding of a sequence of query vectors $\{\mathbf{x}\}$ when the codebook $\mathbf{Y} = \{\mathbf{y_i}\}_{i=1}^{N}$ is given.

Algorithm: Finding the nearest codeword for a query sequence $\{\mathbf{x}\}$ in the k-dimension vector space.

Preprocessing: Evaluate $||\mathbf{y_i}|| = (\sum_{j=1}^{\mathbf{k}} \mathbf{y}_{ij}^2)^{1/2}$ for every codeword $\mathbf{y_i}$ in the codebook

$$\mathbf{Y} = \{\mathbf{y_i}\}_{i=1}^{N}.$$

Sort $\mathbf{Y}$, so that $||\mathbf{y_1}|| \leq ||\mathbf{y_2}|| \leq \cdots | \leq ||\mathbf{y}_N||$.

Steps:

1) Read in an $\mathbf{x}$ which is not encoded yet.
2) Evaluate $2||\mathbf{x}||$.
3) Choose a $\mathbf{y}_{\min}^{(\text{guess})} \in \mathbf{Y}$ and let

$$\mathbf{y}_{\min} = \mathbf{y}_{\min}^{(\text{guess})}; \quad d_{1\min} = d_1\left(\mathbf{x}, \mathbf{y}_{\min}^{(\text{guess})}\right)$$
$$\mathbf{R} = \left\{\mathbf{y_i} \in \mathbf{Y} \;\middle|\; \mathbf{y_i} \neq \mathbf{y}_{\min}^{(\text{guess})}\right\}.$$

4)
   a) Go to 5 if $\mathbf{R}$ is empty.
   b) Choose a $\mathbf{y_i}$ from $\mathbf{R}$.
   c) If $||\mathbf{y_i}||(||\mathbf{y_i}|| - 2||\mathbf{x}||) \geq d_{1\min}$ then do either 4c-i) or 4c–ii).
      i) If $(||\mathbf{y_i}|| \geq ||\mathbf{x}||)$, then delete from $\mathbf{R}$ all $\mathbf{y_j}$ whose $j \geq i$, and go to 4a).
      ii) If $(||\mathbf{y_i}|| \leq ||\mathbf{x}||)$ then delete from $\mathbf{R}$ all $\mathbf{y_j}$ whose $j \leq i$, and go to 4a).
   d) Evaluate $d_1(\mathbf{x}, \mathbf{y_i})$;
      delete $\mathbf{y_i}$ from $\mathbf{R}$;
      if $(d_1(\mathbf{x}, \mathbf{y_i}) \geq d_{1\min})$, then go to 4a).
   e) $d_{1\min} = d_1(\mathbf{x}, \mathbf{y_i})$; (i.e., update the "so far" minimum distortion);
      $\mathbf{y}_{\min} = \mathbf{y_i}$; (i.e., update the "so far" nearest codeword);
      go to 4a).
5) Print out the codeword $\mathbf{y}_{\min}$ because it minimizes (16), and hence, minimizes (15) for the given $\mathbf{x}$.
6) Go to 1.

□

To choose the $\mathbf{y}_{\min}^{(\text{guess})}$ for Step 3 [and the $\mathbf{y_i} \in \mathbf{R}$ for Step 4b)], the authors suggest the use of the codeword in $\mathbf{Y}$ (in $\mathbf{R}$, respectively), whose norm is closest to $||\mathbf{x}||$. Note that the time-saving is obtained through the check done in Step 4c), because Steps 4e) and the time-consuming Step 4d) will then seldom be executed. Also note that in 4c-i) and 4c-ii), we kick out not only $\mathbf{y_i}$ but also many $\mathbf{y_j}$ because

$$||\mathbf{y_j}||(||\mathbf{y_j}|| - 2||\mathbf{x}||) \geq ||\mathbf{y_i}||(||\mathbf{y_i}|| - 2||\mathbf{x}||) \geq d_{1\min}. \tag{20}$$

To see why (20) is true, just note that $f(t) = t(t - 2||\mathbf{x}||)$, as a function of t, is a parabola having the absolute minimum at $t = ||\mathbf{x}||$.

TABLE I
PERFORMANCE COMPARISON AMONG THE ALGORITHMS ON THE TIME (CPU SECONDS) USED TO ENCODE THE LENA IMAGE WHICH IS IN THE TRAINING SET

CODEBOOK SIZE	F.S. CPU TIME	A1 CPU TIME	A2 CPU TIME	A3 CPU TIME	OURS CPU TIME
N=128	29.65	5.07	6.87	4.81	2.53
N=256	72.93	6.65	16.42	6.08	4.25
N=512	145.86	11.82	32.17	10.18	7.36

TABLE II
PERFORMANCE COMPARISON AMONG THE ALGORITHMS ON THE TIME (CPU SECONDS) USED TO ENCODE THE JET IMAGE WHICH IS NOT IN THE TRAINING SET

CODEBOOK SIZE	F.S. CPU TIME	A1 CPU TIME	A2 CPU TIME	A3 CPU TIME	OURS CPU TIME
N=128	29.65	4.47	5.3	4.08	1.89
N=256	72.93	7.99	14.37	6.44	4.15
N=512	145.86	13.7	27.24	10.31	7.23

TABLE III
PERFORMANCE COMPARISON AMONG THE ALGORITHMS ON THE TIME (CPU SECONDS) USED TO ENCODE THE PEPPERS IMAGE WHICH IS NOT IN THE TRAINING SET

CODEBOOK SIZE	F.S. CPU TIME	A1 CPU TIME	A2 CPU TIME	A3 CPU TIME	OURS CPU TIME
N=128	29.65	5.16	6.54	4.65	2.6
N=256	72.93	7.61	16.74	6.49	4.81
N=512	145.86	13.46	32.42	10.7	8.5

IV. EXPERIMENTAL RESULTS

Simulations using standard images are done to evaluate the efficiency of the proposed method. In our experiments, each of the test (or training) 512 × 512 images is divided into non-overlapping 4 × 4 sub-blocks; therefore, each block becomes a 16-dimensional vector. The well-known LBG algorithm [16] is used first to train the image Lena to generate the codebook. After that, the codebook is used to encode the test images such as Jet, Peppers, (and Lena itself). We compared the encoding time of ours with those of the algorithms A1, A2, and A3. Tables I– II show the central processing unit (CPU) time needed for encoding the three different images. The machine used was a Sun-Sparc10 workstation, and the CPU time excluded the overhead (i.e., the preprocessing). Table IV depicts the memory requirement for the algorithms. (Full search algorithm has no preprocessing, and hence, no extra memory was demanded.) Note

TABLE IV
STORAGE COMPARISON AMONG THE ALGORITHMS

METHOD	A1	A2	A3	OURS
STORAGE	$2N^2$	$3N$	$(2\sqrt{k}+1)N$	N

that the vector dimension k equals $4 \times 4 = 16$ in our experiments, whereas N is, as usual, the codebook size. Also note that we had stated in the final paragraph of previous section how to choose $\mathbf{y}_{\min}^{(\text{guess})}$ for our algorithm. As for the $\mathbf{y}_{\min}^{(\text{guess})}$ used for the reported methods, some authors using A1 (see [14]) and the authors of A3 (see [10]) had suggested that if block $\mathbf{B}_{ij}$ is to be coded, then the coding results for four previous blocks $\{\mathbf{B}_{i-1,j-1}; \mathbf{B}_{i-1,j}; \mathbf{B}_{i-1,j+1}; \mathbf{B}_{i,j-1}\}$ are inspected, so that one of them is used as $\mathbf{y}_{\min}^{(\text{guess})}$. We therefore used this kind of $\mathbf{y}_{\min}^{(\text{guess})}$ for A1, A2, and A3.

In order to increase the on-line processing speed of A1 (the TIE-based algrithm), some researchers (e.g., [14]) presort $\{D_{ij} = \|\mathbf{y_i} - \mathbf{y_j}\|^2\}_{j=1}^{N}$ for each $i = 1, 2, \cdots, N$ in the preprocessing procedure to obtain an N-*by*-$(N-1)$ presorted matrix $\tilde{\mathbf{D}}$ (where $\tilde{\mathbf{D}}_{il} \le \tilde{\mathbf{D}}_{i2} \le \tilde{\mathbf{D}}_{i3} \le \cdots \le \tilde{\mathbf{D}}_{i,N-1}$ holds for any $i = 1, \cdots, N$) and an N-*by*-$(N-1)$ Position Matrix $\mathbf{P}$ (e.g. if $\tilde{\mathbf{D}}_{i1} = \tilde{\mathbf{D}}_{i9} \le \tilde{\mathbf{D}}_{i2} = \tilde{\mathbf{D}}_{i7} \le \tilde{\mathbf{D}}_{i3} = \tilde{\mathbf{D}}_{i8} \le \cdots$, then $\mathbf{P}_{i1} = 9, \mathbf{P}_{i2} = 7, \mathbf{P}_{i3} = 8$). We used this presorted version, which required about $2N^2$ storage space, to implement A1.

V. CONCLUSIONS AND DISCUSSIONS

In this study, a kick-out condition using the Cauchy-Schwarz inequality is proposed for the fast codeword searching algorithm. As shown by the experimental results, the proposed method really reduces the CPU time. The low $1N$ memory requirement is also attractive. Furthermore, unlike the A2 algorithm [9], the proposed method has no need to do the extra translation to make all vector components non-negative. The $O(N^2)$ preprocessing time and $O(N^2)$ memory space required by A1 are reduced to $O(NlogO)$ and $O(N)$ in our method. Compared with A3, our method uses only one inequality instead of three; moreover, memory requirement is not only reduced, but also independent of the block size. Due to the simplicity and efficiency, the proposed kick-out condition can either be used alone or be incorporated with some other sophisticated algorithms.

ACKNOWLEDGMENT

The authors wish to thank the referees for their valuable suggestions, especially the comments about the experiments such as the handling of preprocessing time and the choice of $\mathbf{y}_{\min}^{(\text{guess})}$. The authors also thank the editors for prompt processing of the manuscript.

REFERENCES

[1] N. M. Nasrabadi and R. A. King, "Image coding using vector quantization : A review," *IEEE Trans. Commun.*, vol. 36, pp. 957–971, Aug. 1988.
[2] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*: Kluwer Academic Press, 1992.
[3] K. T. Lo and J. Feng, "Predictive mean search algorithms for fast VQ encoding of images," *IEEE Trans. Consumer Electron.*, vol. 41, pp. 327–331, May 1995.
[4] V. Ramasubramanian and K. K. Paliwal, "An efficient approximation-elimination algorithm for fast nearest-neighbour search based on a spherical distance coordinate formulation," *Pattern Recognit. Lett.*, vol. 13, pp. 471–480, July 1992.
[5] Y. C. Lin and S. C. Tai, "Fast feature-based vector quantization algorithm of image coding," *Optical Eng.*, vol. 34, no. 10, pp. 2918–2926, Oct. 1995.
[6] C. D. Bei and R. M. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," *IEEE Trans. Commun.*, vol. COM-33, pp. 1132–1133, Oct. 1985.
[7] M. R. Soleymani and S. D. Morgera, "A fast MMSE encoding technique for vector quantization," *IEEE Trans. Commun.*, vol. COM-37, pp. 656–659, June 1989.
[8] E. Vidal, "An algorithm for finding nearest neighbours in (approximately) constant average time complexity," *Pattern Recognit. Lett.*, vol. 4, pp. 145–157, 1986.
[9] L. Torres and J. Huguet, "An improvement on codebook search for vector quantization," *IEEE Trans. Commun.*, vol. 42, pp. 208–210, Feb./Mar./Apr. 1994.
[10] Y. C. Lin and S. C. Tai, "Fast vector quantization of image coding using integral equations," in *Proc. Nat. Computer Symp.*, Taiwan, R.O.C., Dec. 1995, pp. 729–736.
[11] M. D. Orchard and J. C. Chang, "Fast codebook generation algorithm for vector quantization of images," *Pattern Recognit. Lett.*, vol. 9, pp. 605–609, Dec. 1991.
[12] X. Wu and L. Guan, "Acceleration of the LBG algorithm," *IEEE Trans. Commun.*, vol. 42, pp. 1518–1523, Feb./Mar./Apr. 1994.
[13] K. K. Paliwal and V. Ramasubramanian, "Effect of ordering the codebook on the efficiency of the partial distance search algorithm for vector quantization," *IEEE Trans. Commun.*, vol. 37, pp. 538–540, May 1989.
[14] S. H. Huang and S. H. Chen, "Fast encoding algorithm for VQ-based image coding," *Electron. Lett.*, vol. 26, no. 19, pp. 1618–1619, Sept. 1990.
[15] W. Li and E. Salari, "A fast vector quantization encoding method for image compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 119–123, Apr. 1995.
[16] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84–95, Jan. 1980.