electronics

MDPI

Article

An Efficient Codebook Search Algorithm for Line Spectrum Frequency (LSF) Vector Quantization in Speech Codec

Yuqun Xue [1,2], Yongsen Wang [1,2], Jianhua Jiang [1,2], Zenghui Yu [1], Yi Zhan [1], Xiaohua Fan [1] and Shushan Qiao [1,2,*]

[1] Smart Sensing R&D Center, Institute of Microelectronics of Chinese Academy of Sciences, Beijing 100029, China; xueyuqun@ime.ac.cn (Y.X.); wangyongsen@ime.ac.cn (Y.W.); jiangjianhua@ime.ac.cn (J.J.); yuzenghui@ime.ac.cn (Z.Y.); yizhan@ime.ac.cn (Y.Z.); fanxiaohua@ime.ac.cn (X.F.)

[2] Department of Microelectronics, University of Chinese Academy of Sciences, Beijing 100049, China

* Correspondence: qiaoshushan@ime.ac.cn

Abstract: A high-performance vector quantization (VQ) codebook search algorithm is proposed in this paper. VQ is an important data compression technique that has been widely applied to speech, image, and video compression. However, the process of the codebook search demands a high computational load. To solve this issue, a novel algorithm that consists of training and encoding procedures is proposed. In the training procedure, a training speech dataset was used to build the squared-error distortion look-up table for each subspace. In the encoding procedure, firstly, an input vector was quickly assigned to a search subspace. Secondly, the candidate code word group was obtained by employing the triangular inequality elimination (TIE) equation. Finally, a partial distortion elimination technique was employed to reduce the number of multiplications. The proposed method reduced the number of searches and computation load significantly, especially when the input vectors were uncorrelated. The experimental results show that the proposed algorithm provides a computational saving (CS) of up to 85% in the full search algorithm, up to 76% in the TIE algorithm, and up to 63% in the iterative TIE algorithm. Further, the proposed method provides CS and load reduction of up to 29–33% and 67–69%, respectively, over the BSS-ITIE algorithm.

Keywords: vector quantization (VQ); codebook search; line spectrum frequency (LSF); speech codec

Citation: Xue, Y.; Wang, Y.; Jiang, J.; Yu, Z.; Zhan, Y.; Fan, X.; Qiao, S. An Efficient Codebook Search Algorithm for Line Spectrum Frequency (LSF) Vector Quantization in Speech Codec. *Electronics* **2021**, *10*, 380. https://doi.org/10.3390/electronics10040380

Academic Editor: Manuel Rosa-Zurera

Received: 3 December 2020

Accepted: 2 February 2021

Published: 4 February 2021

1. Introduction

Vector quantization (VQ) is a high-performance technique for data compression. Due to its simple coding and high compression ratio, it has been successfully applied to speech, image, audio, and video compression [1–6]. It is also a key part of the G.729 Recommendation. A major handicap is that a remarkable computational load is required for the VQ of line spectrum frequency (LSF) coefficients of the speech codec [7–12]. Thus, it is necessary to reduce the computation load of VQ.

Conventionally, a full search algorithm is employed to find the code word that is best matched with an arbitrary input vector. However, the full search algorithm demands a large computation load. To reduce the encoding search complexity, many approaches have been proposed [13–28]. These approaches can be classified into four main types, according to their base techniques. The first type is the tree-structured VQ (TSVQ) technique [13–17], the second type is the triangular inequality elimination (TIE)-based approach [18–21], the third type is the equal-average equal-variance equal-norm nearest neighbor search (EEENNS) method [22–24] based on the statistical characteristic values of the input signal, and the last type is the binary search space-structured VQ (BSS-VQ) [25–28] method.

In the TSVQ approaches [13–17], the search complexity can be reduced significantly. However, the reconstructed speech quality is poor because the selected code word is not necessarily the best matched to the input vector. In contrast, the TIE-based method [18–21] can achieve an enormous computational load reduction without loss of speech quality. By

employing the high correlation between the adjacent frames, a TIE method is used to reject the impossible candidate code words. However, the performance of computational load reduction is weakened when the input vectors are uncorrelated. The EEENNS technique employs the statistical characteristics of an input vector, such as the mean, the variance, and the norm, to reject the impossible code words. In order to further reduce the computational load, the BSS-VQ method [25] is proposed, in which the number of candidate code words is closely related to the distribution of the hit probability, and a sharpened distribution at specific code words yields an enormous computational load. However, the probability distribution is not uniform; some subspaces are concentrated, and some are flattened. Thus sometimes the performance is not good.

To solve the above issues, an efficient VQ codebook search algorithm is proposed. Even though the input vectors are uncorrelated, it can still reduce the computational load significantly while maintaining the quantization accuracy. Compared with previous works on the full search algorithm [6], TIE [18], ITIE [21], and the BSS-ITIE [28], the experimental results show that the proposed algorithm can quantize the input vector with the lowest computational load.

The rest of this paper is organized as follows. Section 2 describes the encoding procedure of the LSF coefficients in G.729. Section 3 presents the theory of the proposed algorithm in detail. Section 4 shows the experimental results and performance comparison between the proposed algorithm and previous works. Finally, Section 5 concludes this work.

2. LSF Coefficients Quantization in G.729

The ITU-T G.729 [29] speech codec was selected as the platform to verify the performance of the proposed algorithm. Thus, before introducing the theory of the proposed method, the principle of the LSF quantization in G.729 is introduced here.

The LSF coefficients are obtained by an equation $\omega_i = ar\cos(q_i)$, where q_i is the LSF coefficient in the cosine domain, and w_i is the computed LSF coefficient in the frequency domain.

The procedure of the LSF quantize is organized as follows: a switched 4th moving average (MA) prediction is used to predict the LSF coefficients of the current frame. The difference between the computed and predicted coefficients is quantized using a two-stage vector quantizer. The first stage is a 10-dimensional VQ using a codebook $L1$ with 128 entries (7 bits). In the second stage, the quantization error vectors of the first stage are split into two sub-vectors, which then are quantized by a split VQ associating two codebooks, $L2$ and $L3$, each containing 32 entries (5 bits). Figure 1 illustrates the structure of the two-stage VQ for LSF coefficients.

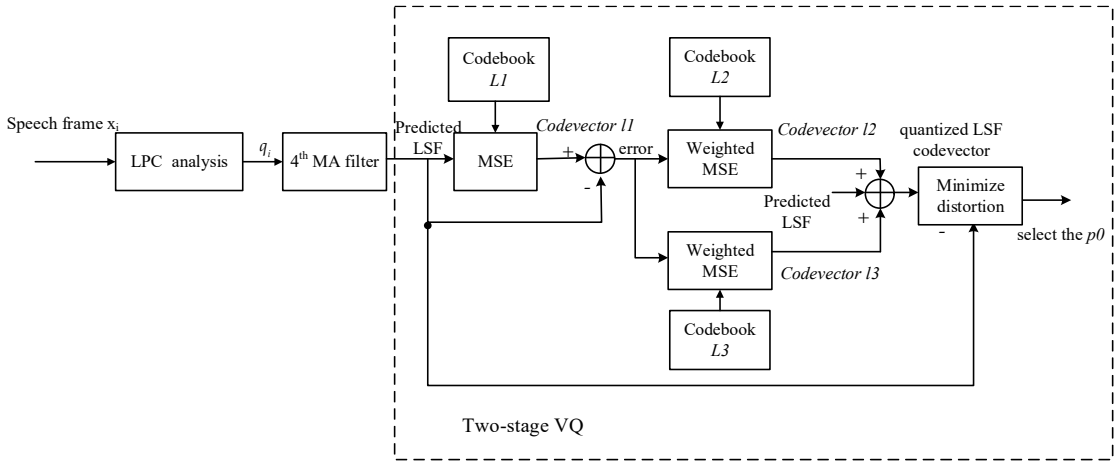

Figure 1. Structure of the two-stage vector quantization (VQ) for line spectrum frequency (LSF) coefficients.

To explain the quantization process, it is convenient to describe the decoding process first. Each quantized value is obtained from the sum of the two code words, as follows:

$$\hat{l}_i = \begin{cases} L1_i(l1) + L2_i(l2) & i = 1,\ldots,5 \\ L1_i(l1) + L3_{i-5}(l3) & i = 6,\ldots,10 \end{cases}, \quad (1)$$

where $l1$, $l2$, and $l3$ are the codebook indices. To guarantee that the reconstructed filters are stable, the vector $\hat{l}_i$ is arranged such that adjacent elements have a minimum distance of $d_{\min}$. This rearrangement process is done twice. First, the quantized LSF coefficients, $\hat{\omega}_i{}^{(m)}$, for the current frame, m, are obtained from the weighted sum of previous quantizer outputs, $\hat{l}_i^{(m-k)}$, and the current quantizer output, $\hat{l}_i^m$.

$$\hat{\omega}_i{}^{(m)} = \left(1 - \sum_{k=1}^{4} \hat{P}_{i,k}\right)\hat{l}_i^m + \sum_{k=1}^{4} \hat{P}_{i,k}\hat{l}_i^{(m-k)} \quad i = 1,\ldots,10. \quad (2)$$

where $\hat{p}_{i,k}$ are the coefficients of the switched MA predictor as defined by parameter, $p0$. For each of the two MA predictors the best approximation to the current LSF coefficients has to be found. The best approximation is defined as the one that minimizes the weighted mean-squared error (MSE).

$$E_{lsf} = \sum_{i=1}^{10} W_i(\omega_i - \hat{\omega}_i)^2. \quad (3)$$

The weights emphasize the relative importance of each LSF coefficient. The weights, W_i, are made adaptive as a function of the unquantized LSF parameters.

$$W_1 = \begin{cases} 1.0 & ,if\ \omega_2 - 0.04\pi - 1 > 0 \\ 10(w_2 - 0.04\pi - 1)^2 + 1\ , \text{ otherwise} \end{cases}. \quad (4)$$

$$W_i,\ for\ 2 \le i \le 9 = \begin{cases} 1.0 & ,if\ \omega_{i+1} - \omega_{i-1} - 1 > 0 \\ 10(\omega_{i+1} - \omega_{i-1} - 1)^2 + 1\ , \text{ otherwise} \end{cases}. \quad (5)$$

$$W_{10} = \begin{cases} 1.0 & ,if\ -\omega_9 + 0.92\pi - 1 > 0 \\ 10(-\omega_9 + 0.92\pi - 1)^2 + 1\ , \textit{otherwise} \end{cases}. \quad (6)$$

Then, the weights ω_5 and ω_6 are multiplied by 1.2 each. The vector to be quantized for the current frame, m, is obtained from Equation (7).

$$I_i^{(m)} = \left[\omega_i^{(m)} - \sum_{k=1}^{4} \hat{p}_{i,k}\hat{l}_i^{(m-k)}\right] \Big/ \left(1 - \sum_{k=1}^{4} \hat{p}_{i,k}\right), i = 1,\ldots,10. \quad (7)$$

The first codebook, $L1$, is searched and the entry, $l1$, that minimizes the unweighted MSE is selected. Then the second codebook, $L2$, is searched by computing the weighted MSE, and the entry $l2$, which results from the lowest error is selected. After selecting the first stage vector, $l1$, and the lower part of the second stage, $l2$, the higher part of the second stage is searched from the codebook, $L3$. The vector, $l3$, that minimizes the weighted MSE is selected. The resulting vector, $\hat{I}_i, i = 1,\ldots,10$, is rearranged twice using the above procedure. The procedure is done for each of the two MA predictors defined by $p0$, and the MA predictor that produces the lowest weighted MSE is selected. Table 1 shows the two groups of coefficients of the MA predictor. When the first group of coefficients is selected, the value of $p0$ is 0. Otherwise, the value of $p0$ is 1. Table 2 shows the bit allocation of the LSF quantizer.

Table 1. The value of the MA predictor coefficients.

$p0$	k	MA Predictor Coefficients									
0	1	0.2570	0.2780	0.2800	0.2736	0.2757	0.2764	0.2675	0.2678	0.2779	0.2647
	2	0.2142	0.2194	0.2331	0.2230	0.2272	0.2252	0.2148	0.2123	0.2115	0.2096
	3	0.1670	0.1523	0.1567	0.1580	0.1601	0.1569	0.1589	0.1555	0.1474	0.1571
	4	0.1238	0.0925	0.0798	0.0923	0.0890	0.0828	0.1010	0.0988	0.0872	0.1060
1	1	0.2360	0.2405	0.2499	0.2495	0.2517	0.2591	0.2636	0.2625	0.2551	0.2310
	2	0.1285	0.0925	0.0779	0.1060	0.1183	0.1176	0.1277	0.1268	0.1193	0.1211
	3	0.0981	0.0589	0.0401	0.0654	0.0761	0.0728	0.0841	0.0826	0.0776	0.0891
	4	0.0923	0.0486	0.0287	0.0498	0.0526	0.0482	0.0621	0.0636	0.0584	0.0794

Table 2. Bit allocation of the LSF quantizer.

Procedure		Code Index	Bits
First stage	10th order	*l1*	7
Second stage	5th low order	*l2*	5
	5th high order	*l3*	5
Selection of predictive filter		*p0*	1
Total			18

3. Proposed Search Algorithm

In this paper, a fast codebook search algorithm of vector quantization is proposed to reduce the computation load of the LSF coefficients' quantization in the G.729 speech codec. Even though the input vectors are uncorrelated, this can still reduce the computation load significantly, while maintaining the quantization accuracy.

In the BSS-VQ method [25], the number of candidate code words is strongly related to the distribution of the hit probability for each subspace, and a sharpened distribution at specific code words yields an enormous computational load while maintaining the quantization accuracy. However, the probability distribution is not uniform: some subspaces are concentrated, and some are flattened. Thus, sometimes its performance is poor. The TIE algorithm [18] uses the strong correlation between adjacent values to narrow the search range. However, its performance is poor when the inputs are uncorrelated.

Considering the drawbacks of the above methods, to further reduce computational load, especially when the inputs are uncorrelated, a novel algorithm is proposed. After the statistical analysis, the code word corresponding to the highest hit probability for each subspace is obtained and selected as the reference code word, and a squared-error distortion look-up table for each subspace is built. The adjacent code word is highly correlated due to the squared-error distortion changing slightly in the squared-error look-up table. The reference code word corresponding to the highest probability is regarded as the best-matched code word. Thus, the smaller the squared-error distortion with the reference code word, the more likely the candidate code word is to be the best matched. Therefore, the TIE technique can be employed to reject the impossible code words. The proposed algorithm consists of the training procedure and encoding procedure. The structure of the proposed algorithm is illustrated in Figure 2, and the theory of the proposed algorithm is presented in detail as follows.

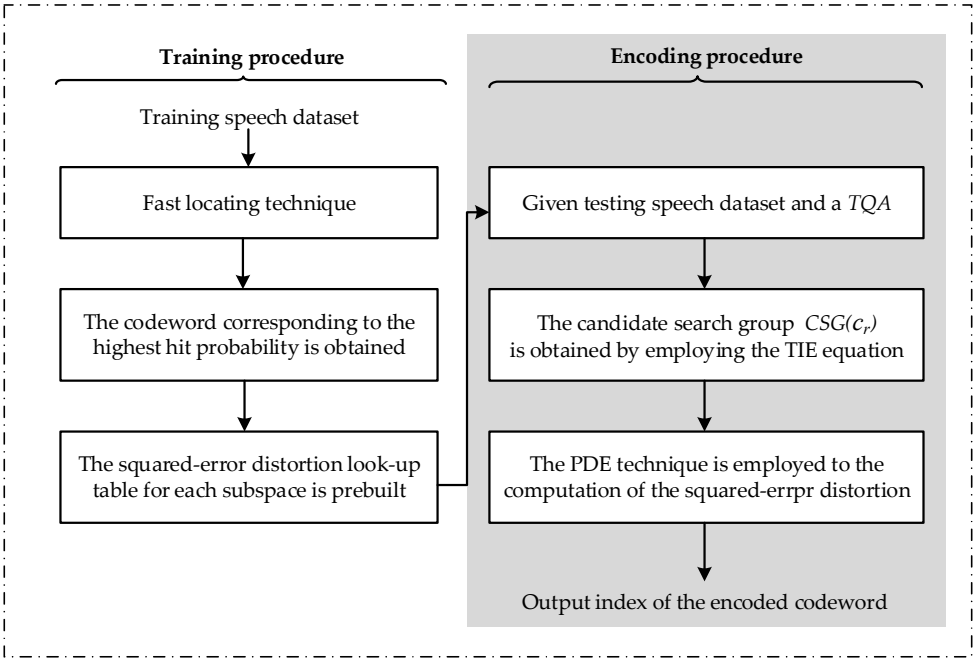

Figure 2. Structure of the proposed search algorithm.

3.1. Training Procedure

In the three-stage training procedure, the squared-error look-up table for each subspace is prebuilt. At stage 1, each dimension is dichotomized into two subspaces. Then, an input vector is assigned to a corresponding subspace according to the entries of the input vector [25]. For instance, when the input is a 10-dimensional vector, there are $2^{10} = 1024$ subspaces in G.729. Before the encoding procedure, a look-up table that contains the hit probability statistical information on the code words is prebuilt.

The dichotomy position for each dimension is defined as the mean of all the code words from the codebook, presented as:

$$mean(k) = \frac{1}{CSize} \sum_{i=1}^{CSize} c_i(k), 0 \le k < Dim, \tag{8}$$

where $c_i(k)$ is the k-th component of the i-th code word c_i, and $mean(k)$ is the average value of all the i-th components. For instance, in the first stage of the LSF coefficients quantization in G.729, $CSize = 128$, $Dim = 10$ in the codebook $L1$. A parameter $v_n(k)$ is defined for vector quantization on the n-th input vector x_n, symbolized as:

$$v_n(k) = \begin{cases} 2^k, x_n(k) \ge mean(k) \\ 0, x_n(k) < mean(k) \end{cases}, 0 \le k < Dim, \tag{9}$$

where $x_n(k)$ is the k-th component of x_n. Then, x_n is assigned to a subspace $j(bss_j)$, where j is the sum of $v_n(k)$ over all the dimensions, presented as:

$$x_n \in bss_j | j = \sum_{k=0}^{Dim-1} v_n(k). \tag{10}$$

For instance, given an input $x_n = \{0.44, 0.08, 0.51, 0.87, 1.26, 1.40, 2.10, 2.18, 2.30, 2.42\}$, $v_n(k) = \{1, 0, 0, 8, 16, 0, 64, 128, 0, 0\}$ for each k, $j = 217$ is obtained by (8) and (9), respectively. Then, the input vector x_n is assigned to the subspace bss_j with $j = 217$. This means that an input vector can be assigned to a corresponding subspace quickly, with only a few basic operations.

At stage 2, the hit probability table for each subspace is prebuilt through a training mechanism, which includes the probability for each code word to be the best-matched code word in each subspace. This is defined as follows: $P_{hit}(c_i|bss_j)$, where $1 \le i \le CSize$, $1 \le j \le Snum$, and $Snum = 2^{Dim}$ is the number of subspaces. Subsequently, the table is sorted in descending order by the hit probability value. For example, when $i = 1$, $P_{hit}(i|bss_j)|_{i=1} = \max_{c_i}\{P_{hit}(c_i)|bss_j\}$ represents the highest hit probability in bss_j and the corresponding code word. The sum of all the hit probabilities for each subspace is 1.0. The hit probability $P_{hit}(c_i|bss_j)$ is computed by Equation (11), and the cumulative probability of the top N code words is symbolized as Equation (12):

$$P_{hit}(c_i|bss_j) = \frac{the\ number\ of\ times\ that\ codeword\ c_i\ falls\ in\ the\ subspace\ bss_j}{the\ total\ number\ of\ times\ that\ all\ the\ candidate\ codewords\ falls\ in\ subspace\ bss_j}, \quad (11)$$

$$P_{cmu}(N|bss_j) = \sum_{i=1}^{N} P_{hit}(n|bss_j), 1 \le N \le CSize, \quad (12)$$

which represents the number of possible candidate code words. Further, to control the quantization accuracy and computational load, a variable named threshold of quantization accuracy (TQA) is defined. This is given a quantity,$N_j(TQA)$, which means the minimum number N that satisfies the equation $P_{cmu}(N|bss_j) \ge TQA$ in a subspace, bss_j, is expressed as:

$$N_j(TQA) = \operatorname{argmin}_N\{N : P_{cum}(N|bss_j) \ge TQA\}, 1 \le N \le CSize, 0 \le j \le Snum, \quad (13)$$

At stage 3, the highest hit probability code word in each subspace is selected as the reference code word c_r, then the squared-error distortion among the reference code word and all the other code words, c_i, in the codebook is calculated by (14).

$$d(c_r, c_i) = \|c_r - c_i\| = \sum_{i=1}^{k-1} (c_r - c_i)^2 \quad (14)$$

Subsequently, the squared-error distortion look-up table for each subspace is built and then is sorted in ascending order. Algorithm 1 shows the pseudo-code of the training procedure.

Algorithm 1. Training procedure of the proposed algorithm

Step 1. Give a training speech dataset and a value, *TQA*.
Step 2. The input vector is assigned to a corresponding subspace, bss_j, by (9) and (10).
Step 3. Repeat Step 2 until all the input vectors are encoded.
Step 4. The hit probability table is obtained by (11) and (12) and then sorted in descending order.
Step 5. Select the highest hit probability code word as the reference code word, the squared-error distortion among the reference code word and other code words, c_i, in the codebook is calculated by (14).
Step 6. The squared-error distortion look-up table is prebuilt for each subspace and is sorted in ascending order.

3.2. Encoding Procedure

Given a testing speech set and a value for TQA, in step 1, an input vector is assigned to a search subspace by (9) and (10). In step 2, a triangular inequality elimination (TIE) formulation is used to reject the impossible candidate code words [18]. If $d(c_r, c_i) > 4d(c_r, x)$, then $d(c_i, x) > d(c_r, x)$, thus the computation of $d(c_i, x)$ is eliminated. Where x is the input vector, c_r is the reference code word. A group composed of all the code words, c_i, which

satisfy the above conditions is selected, then a group defined as a candidate search group (*CSG*) is built, and the number of code words in the CSG is symbolled as $N(c_r)$.

$$\text{TIE}: \ CSG(c_r) \ = \ \{c_i | d(c_r, c_i) < 4d(c_r, x)\}, 1 \leq i \leq CSize. \tag{15}$$

In step 3, after the $CSG(c_r)$ is obtained, the squared-error distortion between the input vector and each candidate code word is computed. The best-matched code word is the one that makes the squared-error distortion between the input vector, x_n, and the candidate code word, c_i, minimized.

As stated by BEI [30], a minimum squared-error distortion computation method called partial distortion elimination (PDE) is employed to reduce the number of multiplication operations. This method can decide whether the current code word is the best matched or not before the whole squared-error distortion is calculated. The pseudo-code of the PDE method is shown in Algorithm 2, where $CSize$, Dim, $d_{\min}$, $C(i, j)$ are the codebook size, the dimension of the input vector, the minimum distortion value, and the code word in the codebook, respectively. After the abovementioned description, the encoding procedure of the proposed algorithm can be summarized as Algorithm 3.

Algorithm 2. Pseudo-code of the PDE algorithm

```
1 j = 0, d_min = 28
2 whilej < CSize
3 dist = 0
4 i = 0
5 while (dist < d_min)
6 tmp = x(j) − C(i, j)
7 dist = dist + tmp ∗ tmp
8 i = i + 1
9 end
10 if ((i = = Dim) && (dist < d_min))
11 d_min = dist
12 index = j
13 end
14 j = j + 1
15 end
```

Algorithm 3. Encoding procedure of the proposed algorithm.

Step 1. Given a testing speech set and a value for TQA.
Step 2. The input vector is quickly assigned to a corresponding subspace, bss_j, by (9) and (10).
Step 3. The number of candidate code words, $N_k(TQA)$, is found directly from the prebuilt squared-error distortion look-up table for each subspace.
Step 4. The code word corresponding to the highest hit probability is selected as a reference, c_r, compute $d(c_r, x)$, and then the $CSG(c_r)$ and $N(c_r)$ are obtained by (15).
Step 5. Starting at $k = 1$, the $d(c_r, c_k)$ is obtained directly from the squared-error distortion look-up table.
Step 6. If $(d(c_r, c_k) < 4d(c_r, x))$, then compute $d(c_k, x)$ by Algorithm 2, $k \ = \ k + 1$; repeat Step 4 until $k \ = \ N(c_r)$. Then the index of the best-matched code word is obtained.
Step 7. Output index of the best-matched code word.
Step 8. Repeat Steps 2–6 until all the input vectors are encoded.

4. Experiment and Results

4.1. Experimental Environment

Here, the first-stage quantization procedure of LSF coefficients in G.729 was selected as a platform to illustrate the performance of the proposed algorithm. The first-stage codebook included 128 code words, and each code word was a 10-dimensional vector. There were 1024 subspaces. The Aurora speech dataset [31] was used as the training and testing speech data. There were 1001 clean speech files and 1001 speech files with noise

from Test-A set, spoken by 50 males and 50 females used in this paper. The testing speech signals were sampled at 8 kHz with a resolution of 16 bits per sample.

4.2. Selection of the Training Dataset

As introduced in Section 3, the proposed method included a training procedure and an encoding procedure. The training procedure provided a squared-error look-up table for the encoding procedure. The selection of the training speech dataset directly affected the application scope of the squared-error look-up table. Further, it affected the computation loads and quantization accuracy of the subsequent encoding process. Therefore, we discuss the influence of the selection of the training dataset on the application scope of the squared-error look-up table.

The Aurora Test-A set included 1001 clean files and 1001 noisy speech files that were used to train and test the robustness of the proposed algorithm. Here we will compare the performance of the proposed method with three experimental environments. To evaluate the quantization accuracy of the proposed algorithm, a parameter defined as error rate (ER) was proposed, symbolized as $ER = \frac{\text{uncorrected quantized frames}}{\text{total input frames}}$. The average search times are symbolized as ASN, and the reduction of the ASN is presented as a computational saving (CS) and symbolized as $CS = \frac{ASN_1 - ASN_2}{ASN_1}$, given to evaluate the reduction of the computational load. The following three experimental results were all obtained with the value $TQA = 0.99$.

For the first experimental conditions, the 1001 clean files, which included 350,866 speech frames, were all selected as the training set, then a squared-error look-up table which is symbolized as table A was obtained after the training procedure. Then, table A was employed in the encoding process. The testing datasets were clean files and noisy speech files, respectively. When there were 500 clean files, which included 174,630 speech frames used as the testing dataset, the experimental results show that there were 1412 uncorrected quantized frames, thus ER = 0.81%, and ASN = 18.82. When there were 201 speech files with noise, which included 70,170 frames used as a testing dataset, the experimental results show that there were 2579 uncorrected quantized frames, thus t ER = 3.7% and ASN = 18.9.

For the second experimental conditions, the training dataset included 500 clean speech files and 500 speech files with noise, which included 348,724 frames. Then, when the training dataset was used as a testing dataset, the experimental results show there were 1062 uncorrected quantized frames, thus $ER = 0.7\%$ and ASN = 18.8. When the other 201 clean speech files, which included 70,170 frames used as testing data, the experimental result shows that there were 2038 uncorrected quantized frames, thus $ER = 2.9\%$ and ASN = 19.3. When the other 201 speech files with noise, which included 70,170 frames used as a testing set, the experimental results show that there were 1975 uncorrected quantized frames, thus $ER = 2.8\%$ and ASN = 18.9.

For the third experimental conditions, the training dataset included 1001 clean files and 1001 speech files with noise, which included 701,508 frames. When there were 201 clean speech files used as a testing set, the experimental results show that there were 556 uncorrected quantized frames, thus $ER = 0.79\%$ and ASN = 20. When there were 201 speech files with noise, which included 70,170 frames used as a testing set, the experimental results show that there were 605 uncorrected quantized frames, thus $ER = 0.8\%$ and ASN = 19.9.

Details of the above three experiments and the corresponding experimental results can be found in Table 3. Even under the worst training conditions, 1, where the training set was clean speech and the testing set was speech files with noise, the experimental results show that $ER = 3.7\%$ and ASN = 18.9. Under the best training conditions, 3, the experimental results show that $ER = 0.8\%$ and ASN = 20. Comparing the results of these three conditions, the ER value ranges from 0.7% to 3.7%, and the ASN value ranges from 18.82 to 20. It can be concluded that when the training set was large enough, the selection of the training set had no significant influence on the encoding process. The robustness of the prebuilt squared-error look-up table was good. The variation ranges of ER and ASN were within acceptable limits.

Table 3. The selection of the training dataset.

Experimental Conditions	Training Set	Testing Set	*ER*	ASN
Conditions 1	1001 clean files	500 clean files	0.81%	18.82
		201 files with noise	3.7%	18.9
Conditions 2	500 clean and 500 files with noise	Training set	0.7%	18.8
		201 clean files	2.9%	19.3
		201 files with noise	2.8%	18.9
Conditions 3	1001 clean and 1001 files with noise	201 clean files	0.79%	20
		201 files with noise	0.8%	19.9

4.3. Performance of the Proposed Method

Here we choose experiment 1 to illustrate the performance of the proposed method. The generation of the squared-error look-up table is a very important process of the proposed algorithm. Thus, we extracted some intermediate experimental data as examples to illustrate the created procedure. Figure 3 illustrates the design procedure of the squared-error look-up table of the proposed algorithm. It shows that the generation of the squared-error look-up table can reduce the number of candidate code words significantly and reduce the search range.

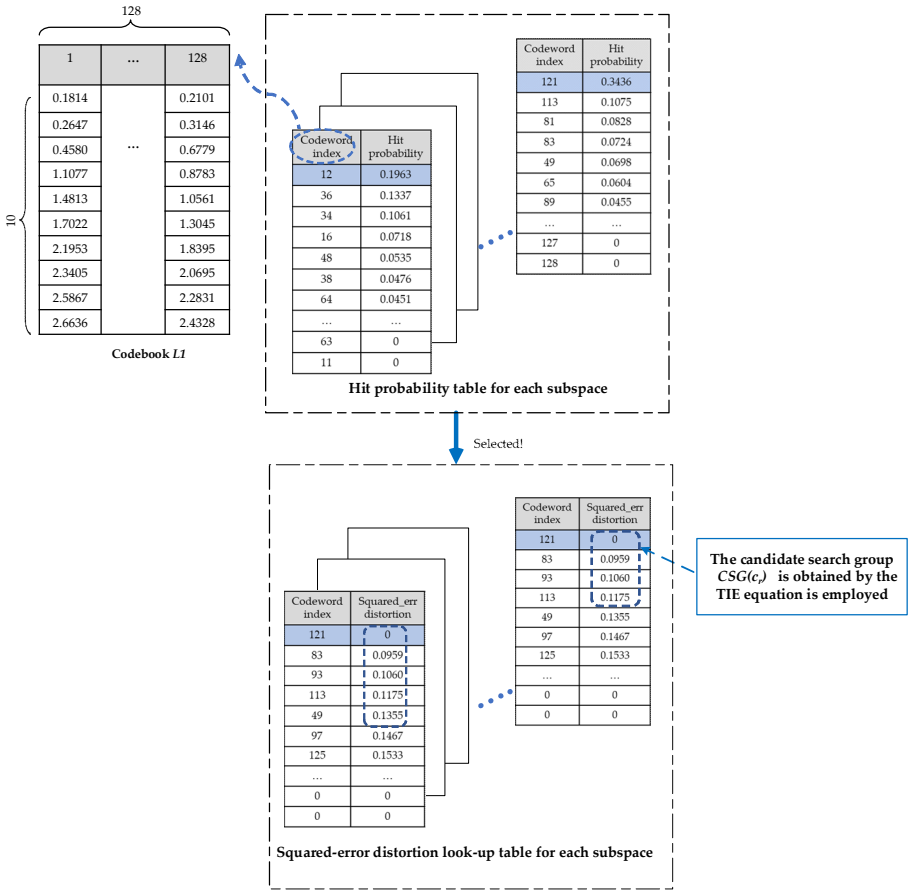

Figure 3. The squared-error distortion look-up table is designed for each subspace.

The computational load and the quantization accuracy are compared for the proposed algorithm with TIE [18], ITIE [21], and BSS-ITIE [28] approaches. With the performance of the full search algorithm as the benchmark, Table 4 gives the comparison of *ER*, ASN, and

CS for the proposed algorithm with TIE [18], ITIE [21], and BSS-ITIE [28] approaches. The experimental results show that the proposed algorithm provided *CS* of up to 92% when $TQA = 0.90$ and when $TQA = 0.99$, it still reduced the computational load by 85%. Compared to the TIE and ITIE methods, the proposed method provided *CS* of up to 76% and 63% with almost the same quantization accuracy.

Table 4. Comparison of the computational load and ER of various methods.

Methods		*ER*	ASN	*CS*
Full search algorithm		0	128	Benchmark
TIE		0	78.37	38.77%
ITIE		0	51.21	60%
BSS-ITIE (*TQA*)	0.90	13.85%	10.39	91.88%
	0.91	13.10%	10.84	91.53%
	0.92	12.20%	11.35	91.13%
	0.93	11.32%	11.88	90.72%
	0.94	10.38%	12.54	90.20%
	0.95	9.28%	13.28	89.63%
	0.96	8.28%	14.11	88.98%
	0.97	7.02%	15.23	88.10%
	0.98	5.97%	16.53	87.09%
	0.99	4.66%	18.47	85.57%
Proposed (*TQA*)	0.90	7.89%	10.16	92.06%
	0.91	7.15%	10.60	91.72%
	0.92	6.37%	11.10	91.33%
	0.93	5.65%	11.65	90.90%
	0.94	4.86%	12.33	90.37%
	0.95	4.08%	13.10	89.77%
	0.96	3.35%	13.98	89.08%
	0.97	2.47%	15.17	88.15%
	0.98	1.67%	16.61	87.02%
	0.99	0.81%	18.82	85.30%

To further evaluate the reduction of computational load, the comparison of the average number of basic operations, including addition, multiplication, and comparison, is shown in Table 5 and is illustrated as a bar graph in Figure 4. The multiplication operation was the dominant computation, with the highest computational complexity. The reduction in the number of multiplications is the load reduction (LR), symbolized as $LR = \frac{\text{MulN}_1 - \text{MulN}_2}{\text{MulN}_1}$, where MulN is the number of multiplications. The proposed algorithm provided *LR* up to 90% in the full search algorithm, with almost the same quantization accuracy.

Table 5. Comparison of the basic operation numbers between the proposed method and other methods.

Methods	*TQA*	Additions	MulN	Comparisons	*LR*
Full search algorithm	-	2560	1280	128	Benchmark
TIE	-	1654	794	206	38%
ITIE	-	978	628	369	51%
BSS-ITIE	0.99	379	291	106	77%
Proposed	0.99	518	125	50	90%
	0.95	387	95	37	92.5%
	0.94	369	91	36	92.8%

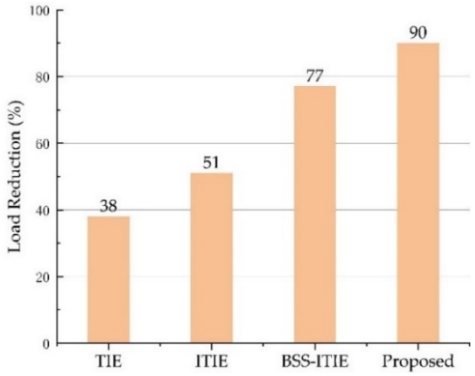

Figure 4. Bar graph representation of Table 4.

Table 6 and Figure 5 give the comparison results with the BSS-ITIE [28] as the benchmark. When $TQA = 0.99$, the ASN of the BSS-ITIE algorithm was 18.47 with $ER = 4.66\%$. In comparison, the ASN of the proposed algorithm was 18.82 with $ER = 0.81\%$, and $LR = 57\%$. This indicates the proposed algorithm can obtain a better speech quality than the BSS-ITIE algorithm with great reduction of the number of multiplications. On the other hand, when $TQA = 0.95$, the ASN of the proposed algorithm was 13.10 with $ER = 4.08\%$. When $TQA = 0.94$, the ASN of the proposed algorithm was 12.33 with $ER = 4.86\%$. This indicates the proposed algorithm provided CS of about 29–33%, and LR up to 67–69%, over the BSS-ITIE algorithm with almost the same ER.

Table 6. Computational savings comparison between the BSS-ITIE [28] method and the proposed algorithm.

Methods	*TQA*	*ER*	ASN	MulN	*LR*	*CS*
BSS-ITIE	0.99	4.66%	18.47	291	Benchmark	
Proposed	0.99	0.81%	18.82	125	57%	0
	0.95	4.08%	13.10	95	67%	29%
	0.94	4.86%	12.33	91	69%	33%

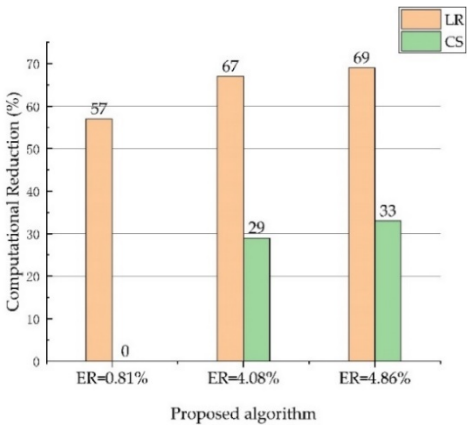

Figure 5. Bar graph representation of Table 6.

Figure 6 shows that when ASN was about equal to 19, the ER of the proposed method was equal to 0.81%, while with the BSS-ITIE it was equal to 4.66%. For instance, when ASN was approximately equal to 13, the ER of the proposed method was lower than that of the BSS-ITIE method by about 5%. When ER was approximately equal to 4%, the ASN of the proposed method was about 5.5 lower than that of the BSS-ITIE method. Thus, the proposed algorithm had a significantly better performance than the BSS-ITIE method.

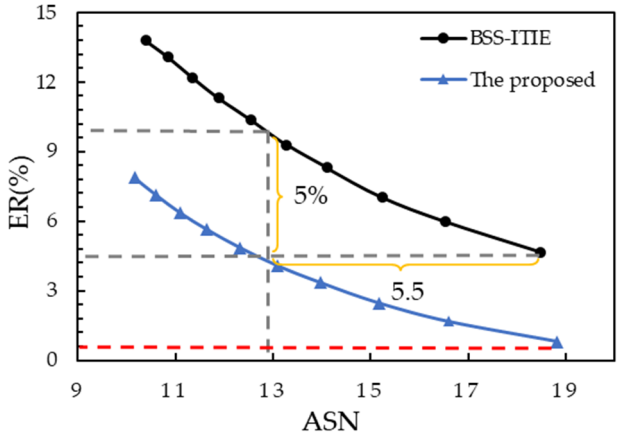

Figure 6. Comparison of the *ER* and ASN between the BSS-TIIE [28] method and the proposed algorithm.

In addition, to better measure the quantization error, the average vector quantization error (AVQR) was defined as the absolute error value between the quantized code word and the best-matched code word. The *AVQR* was computed by Equation (16).

$$AVQR = \frac{\sum_{i=1}^{i=L} abs(\hat{c}_i - c_i)}{L}, \quad (16)$$

where $\hat{c}_i$ was the quantized code word, and c_i was the best-matched code word with the input vector which was searched by the full search algorithm. L was the total number of input speech frames. The *AVQR* value of the BSS-ITIE [22] and the proposed algorithm were computed, while the *TQA* ranged from 0.90 to 0.99, respectively. Table 7 shows the *AVQR* comparison between the BSS-ITIE [22] method and the proposed method. The experimental results show that all the *AVQR* values of the proposed method were lower than 0.1, and the *AVQR* value of the BSS-ITIE method ranged from 0.0695 to 0.2324. Further, the max value of *AVQR* for the proposed method was 0.0974 when *TQA* = 0.90, which is about equal to the *AVQR* value of the BSS-ITIE method when *TQA* = 0.98. Thus, the experimental results show that the proposed method can obtain a much lower quantization error than the BSS-ITIE method.

Table 7. Average vector quantization error comparison between the BSS-ITIE [28] method and the proposed method.

TQA	BSS-ITIE	Proposed
0.90	0.2324	0.0974
0.91	0.2166	0.0872
0.92	0.1993	0.0764
0.93	0.1821	0.0673
0.94	0.1648	0.0572
0.95	0.1450	0.0475
0.96	0.1273	0.0386
0.97	0.1074	0.0278
0.98	0.0905	0.0185
0.99	0.0695	0.0089

5. Conclusions

In this paper, an efficient codebook search algorithm for the VQ of the LSF coefficients is proposed to reduce the computation load. A squared-error look-up table was prebuilt in the training procedure and then the encoding procedure began. An input vector was quickly assigned to a search subspace, then the CSG was obtained by employing the

TIE equation. Subsequently, a PDE technique was employed to reduce the number of multiplications. The experimental results show that the proposed algorithm provided a *CS* of up to 85% in the full search algorithm, up to 76% in the TIE algorithm, and 63% in the iterative TIE (ITIE) algorithm when $TQA = 0.99$. Compared to the BSS-ITIE algorithm, the proposed method provided a *CS* and *LR* of up to 29–33% and 67–69%, respectively, with almost the same quantization accuracy. Further, a trade-off between the computation loads and quantization accuracy could easily be made to meet a user's requirement when performing VQ encoding. This work would be beneficial for reaching the energy-saving requirement when implemented in a speech codec of mobile devices, and the reduction of computation load is helpful for the G.729 Recommendation's application in real-time speech signal processing systems.

Author Contributions: Y.X. developed the idea and wrote this article. Y.W., Z.Y., J.J., Y.Z., X.F. and S.Q. put forward some constructive suggestions for revision. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: I would like to thank the Smart Sensing R&D Center, Institute of Microelectronics of the Chinese Academy of Sciences, which supported us in this work.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

VQ	Vector quantization
LPC	Linear prediction coding
LSF	Line spectrum frequency
BSS-VQ	binary space search vector quantization
CSG	candidate search group
TQA	threshold of quantization accuracy
TIE	triangular inequality elimination
ITIE	iterative triangular inequality elimination
PDE	partial distortion elimination
ASN	average search numbers
CS	computational saving
LR	load reduction
CS-ACELP	conjugate-structure algebraic-code-excited linear prediction
EEENNS	equal-average equal-variance equal-norm nearest neighbor search
AVQR	average vector quantization error.

References

1. Cheng, Z.; Yang, F.; Gao, B. Partial Discharge Pattern Recognition of XLPE Cable Based on Vector Quantization. *IEEE Trans. Magn.* **2019**, *55*, 1–4. [CrossRef]
2. Lu, X.; Wang, H.; Dong, W. Learning a Deep Vector Quantization Network for Image Compression. *IEEE Access* **2019**, *7*, 118815–118825. [CrossRef]
3. Wu, Z.-B.; Yu, J. Vector quantization: A review. *Front. Inf. Technol. Electron. Eng.* **2019**, *20*, 507–524. [CrossRef]
4. Gray, R.M. Vector quantization. *IEEE ASSP Mag.* **1984**, *1*, 4–29. [CrossRef]
5. Makhoul, J.; Roucos, S.; Gish, H. Vector Quantization in Speech Coding. *Proc. IEEE* **1985**, *73*, 1551–1588. [CrossRef]
6. Linde, Y.; Buzo, A.; Gray, R.M. An Algorithm for Vector Quantizer Design. *IEEE Trans. Commun.* **1980**, *28*, 84–95. [CrossRef]
7. Mobini, N.; Vahdat, B.; Radfar, M.H. An FPGA based implementation of G.729. In Proceedings of the IEEE International Symposium on Circuits and Systems, Kobe, Japan, 23–26 May 2005.
8. Hwang, S.-H. Computational improvement for G.729 standard. *Electron. Lett.* **2000**, *36*, 1163–1164. [CrossRef]
9. Sheikh, N.M.; Siddiqui, K.I. Real-time implementation and optimization of ITU-T's G.729 speech codec running at 8 kbits/sec using CS-ACELP on TM-1000 VLIW DSP CPU. In Proceedings of the IEEE International Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century, Lahore, Pakistan, 30 December 2001.
10. Chen, C.; Fu, X. G.729/A Speech Coder: Multichannel TMS320C62x Implementation. 2000. Available online: https://www.ti.com/lit/an/spra564b/spra564b.pdf?ts=1612342823282&ref_url=https%253A%252F%252Fwww.google.com.hk%252F (accessed on 3 February 2021).

11. Swee, L.H. Implementation of G.729 on the TMS320C54x. 2000. Available online: https://www.ti.com/lit/an/spra656/spra656.pdf (accessed on 3 February 2021).
12. Zhang, Z.; Wang, G.; Zhang, Y. Implementation of G.729 algorithm based on ARM9. In Proceedings of the 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), Yichang, China, 21–23 April 2012.
13. Chang, R.F.; Chen, W.T.; Wang, J.S. Image sequence coding using adaptive tree-structured vector quantisation with multipath searching. *IEEE Proc. I Commun. Speech Vis.* **1992**, *139*, 9–14. [CrossRef]
14. Chou, P.A.; Lookabaugh, T.; Gray, R.M. Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Trans. Inf. Theory* **1989**, *35*, 299–315. [CrossRef]
15. Balakrishnan, M.; Pearlman, W.A.; Lu, L. Variable-rate tree-structured vector quantizers. *IEEE Trans. Inf. Theory* **1995**, *41*, 917–930. [CrossRef]
16. Yang, S.-B. Variable-branch tree-structured vector quantization. *IEEE Trans. Image Process.* **2004**, *13*, 1275–1285. [CrossRef] [PubMed]
17. Chu, C.-P.; Hwang, S.-H.; Chang, S.-C. An improvement to tree-structured vector quantization. *Appl. Mech. Mater.* **2013**, *284–287*, 2926–2929. [CrossRef]
18. Huang, S.-H.; Chen, S.-H. Fast encoding algorithm for VQ-based image coding. *Electron. Lett.* **1990**, *26*, 1618–1619. [CrossRef]
19. Hsieh, C.-H.; Liu, Y.-J. Fast search algorithms for vector quantization of images using multiple triangle inequalities and wavelet transform. *IEEE Trans. Image Process.* **2000**, *9*, 321–328. [CrossRef]
20. Huang, C.-J.; Yeh, C.-Y.; Hwang, S.-H. An improvement of the Triangular Inequality Elimination Algorithm for Vector Quantization. *Appl. Math. Inf. Sci.* **2015**, *9*, 229–235. [CrossRef]
21. Yeh, C.-Y. Iterative Triangular Inequality Elimination Algorithm for Codebook Search of Vector Quantization. *IEEJ Trans. Electr. Electron. Eng.* **2018**, *13*, 1528–1529. [CrossRef]
22. Ra, S.-W.; Kim, J.-K. A fast mean-distance-ordered partial codebook search algorithm for image vector quantization. *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.* **1993**, *40*, 576–579. [CrossRef]
23. Lu, Z.M.; Sun, S.H. Equal-average equal-variance equal-norm nearest neighbor search algorithm for vector quantization. *IEICE Trans. Inf. Syst.* **2003**, *86*, 660–663.
24. Chen, S.X.; Li, F.W. Fast encoding method for vector quantisation of image using subvector characteristics and Hadamard transform. *IET Image Process.* **2011**, *5*, 18–24. [CrossRef]
25. Lin, T.-H.; Yeh, C.-Y.; Hwang, S.-H.; Chang, S.-C. Efficient binary search space-structured VQ encoder applied to a line spectral frequency quantisation in G.729 standard. *IET Commun.* **2016**, *10*, 1183–1188. [CrossRef]
26. Yeh, C.-Y. An Efficient VQ Codebook Search Algorithm Applied to AMR-WB Speech Coding. *Symmetry* **2017**, *9*, 54. [CrossRef]
27. Yeh, C.-Y. An efficient search algorithm for ISF vector quantization in AMR-WB speech codec. *IEEJ Trans.* **2016**, *11*, 829–831. [CrossRef]
28. Yeh, C.-Y.; Huang, H.-H. An Upgraded Version of the Binary Search Space-Structured VQ Search Algorithm for AMR-WB Codec. *Symmetry* **2019**, *11*, 283. [CrossRef]
29. ITU-T Recommendation. *G.729: Coding of Speech at 8 kbit/s Using Conjugate-Structure Algebraic-Code-Excited Linear Prediction (CS-ACELP)*; International Telecommunication Union: Geneva, Switzerland, 1996.
30. Bei, C.-D.; Gray, R.M. An improvement of the minimum distortion encoding algorithm for vector quantization. *IEEE Trans. Commun.* **1985**, *33*, 1132–1133.
31. Aurora Corpus. Available online: http://portal.elda.org/en/catalogues/free-resources/free-lrs-set-1/ (accessed on 15 November 2020).