

國立交通大學

電信工程學系

碩士論文

CELP 降頻取樣碼簿搜尋與修改碼簿

Downsample Codebook Search
with Modified Codebook in CELP

研究生：蕭聖穎

指導教授：謝世福 博士

中華民國 九十四 年 七 月

Contents

Chinese Abstract	i
English Abstract	ii
Acknowledgement	iii
List of Figures	iv
List of Tables	v
Chapter 1 Introduction	1
Chapter 2 Code Excited Linear Prediction (CELP) and FS1016	3
2.1 Basic CELP Structure	4
2.2 The FS1016 Standard.....	7
2.2.1 Overview of Standard	7
2.2.2 The FS1016 structure.....	9
2.2.3 Adaptive Codebook	16
2.2.4 Stochastic codebook.....	19
Chapter 3 Downsample Codebook Search with Modified Codebook	23

3.1	Downsample in Match Calculation.....	25
3.2	Modified Codebook	29
3.3	Re-search method to increase hit rate	43
3.4	Classification before Stochastic Codebook Search.....	48
Chapter 4 Simulation Results		54
4.1	Comparison in Downsample two, three and four	55
4.1.1	Downsample Two	56
4.1.2	Downsample Three	57
4.1.3	Downsample Four.....	58
4.1.4	Discussion	59
4.2	Comparison between different proposed method	60
Chapter 5 Conclusion and Future work		66
Bibliography		68



在 CELP 中的降頻取樣碼簿搜尋 與修改碼簿

學生：蕭聖穎

指導教授：謝世福

國立交通大學電信工程學系碩士班



CELP(碼簿激發線性預測)是一種利用 ABS(合成分析法)而可將語音壓縮到低位元率的演算法。但是，其高運算量在實際應用上總是個問題，而其高運算量主要是在合成分析法所構成的封閉迴圈中。在此論文，我們提出降頻取樣碼簿搜尋與修改碼簿來減少在 FS1016 裡減少封閉迴路的搜尋的運算量。然而，這個方法會導致音質的下降。所以，我們又提出再搜尋及分類法來改善音質變差這個缺點。經過模擬，我們所提出的方法可以大大降低運算量，同時，所呈現的音質非常接近 FS1016。

Downsample Codebook Search with Modified Codebook in CELP

Student: S. Y. Hsiao

Advisor: S.F. Hsieh

Department of Communication Engineering
National Chiao Tung University

Abstract



CELP is a low bit rate speech compression algorithm that applies the ABS (analysis-by-synthesis) method. However, the high computation is always an issue in applications, and the main computation lies in the close-loop in the ABS. In this thesis, we propose the downsample method and modified codebook to cut down the computation in the close-loop search based on FS1016 standard. However, this method would suffer a little degradation in speech quality. Thus, the re-search and classifying method has adopted as a remedy. Finally, the proposed method can reduce the computational load drastically, but the speech quality is nearly the same with standard FS1016.

Acknowledgement

本論文及這兩年碩士學業能夠順利完成，要感謝我的指導教授謝世福老師。老師耐心的指正，讓我受益良多。而老師在研究上的嚴謹的思維及深度的見解，能讓我在學習上能夠更加謙虛。

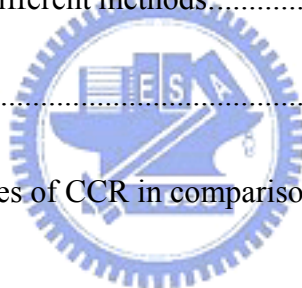
感謝在這兩年來陪伴我的實驗室的學長，同學和學弟。也感謝好朋友及我的女朋友立玫的支持和鼓勵。

最後，要感謝家人在困境中依然支持我唸書，讓我能夠在這兩年能無後顧之憂的學習，由衷的感謝我的家人。



List of Table

Table 2-1	Bit allocation in FS1016 encoder.....	8
Table 3-1	The probability of hit in $l = 10, 20, 30, 40, 50, 60$	29
Table 3-2	Computation load in classification, standard search and downsample along with re-search.....	53
Table 4-1	Applied method in different situation.....	60
Table 4-2	The SNR(dB) in different methods.....	62
Table 4-3	CCR scale.....	62
Table 4-4	The averaged scores of CCR in comparison of situation A (standard).....	63

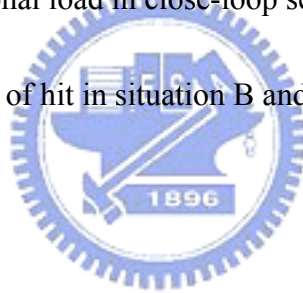


List of Figure

Figure 2-1	Basic CELP structure	5
Figure 2-2	Separate the short-term filter response to zero-input response and zero-state response.	10
Figure 2-3	Separate the long-term filter to zero-input response and zero-state response further.	12
Figure 2-4	The structure in FS1016.	13
Figure 2-5	The structure of FS1016 decoder.	15
Figure 2-6	Close-loop search structure with zero-input long-term filter.	17
Figure 2-7	The adaptive codebook search structure.	18
Figure 2-8	Overlapping structure in stochastic codebook.	20
Figure 2-9	The close-loop structure of stochastic codebook.	21
Figure 3-1	The space diagram.	27
Figure 3-2	The probability of hit in \mathbb{R}^l , where $l = 10, 20, 30, 40, 50, 60$.	28
Figure 3-3.	The close-loop search incorporates with downsampling search.	30
Figure 3-4	Exchange the filter and downsampler.	31

Figure 3-5	Turning IIR filter to FIR filter.	32
Figure 3-6	Moving downsample to the front-end of the filter.	34
Figure 3-7	Polyphase structure with downsample two.	35
Figure 3-8	The rearrangement of polyphase component.	37
Figure 3-9	Rearranged polyphase structure in downsampling by 2.	38
Figure 3-10	The comparison between original codebook and modified codebook.	40
Figure 3-11	Rearranged polyphase structure in downsampling by three with one overlapping branch.	41
Figure 3-12	Rearranged polyphase structure in downsampling by three with two overlapping branch.	42
Figure 3-13	Probability of rank which $m_k^{(30)}$ fall in with the k chosen in dimension 60.	45
Figure 3-14	The spectrum of $\mathbf{s}_{3,k}$ when “hit” or “miss”.	49
Figure 3-15	Flow chart of classification.	52
Figure 4-1	Probability of rank which $m_k^{(30)}$ fall in where the k is chosen in dimension 60.	56
Figure 4-2	The spectrum of $\mathbf{s}_{3,k}$ when “hit” or “miss” in downsampling 2.	56

Figure 4-3	Probability of rank which $m_k^{(20)}$ fall in where the k is chosen in dimension 60.	57
Figure 4-4	The spectrum of $\mathbf{s}_{3,k}$ when “hit” or “miss” in downsampling 3.	57
Figure 4-5	Probability of rank which $m_k^{(15)}$ fall in where the k is chosen in dimension 60.	58
Figure 4-6	The spectrum of $\mathbf{s}_{3,k}$ when “hit” or “miss” in downsampling 4.	58
Figure 4-7	The SNR in different methods.	61
Figure 4-8	The computational load in close-loop search.	64
Figure 4-9	The probability of hit in situation B and D.	65



Chapter 1

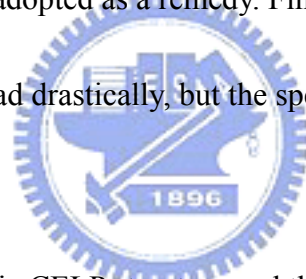
Introduction

The code excited linear prediction (CELP) is widely used in speech compression to achieve the low bit rate ranging from 4.8 kbit/s to 16 kbit/s but still has a good quality. The core technique that CELP uses is analysis-by-synthesis which forms a close-loop that incorporates codebook with synthesis-filter to find the best excitation source in the codebook. This technique works well in low bit rate compression but is computationally intensive. Thus, this is often a bottleneck in the practical implementations. Consequently, many research efforts focus on the reduction in the computational load in the close-loop search.

So far, many methods which aim at reducing the computation in the close-loop search have been proposed, and most of them have been adopted in standard. For example, the overlapping codebook structure and ternary-valued samples are

utilized to in standard FS1016. Besides, VCELP which relies on basis vectors to generate the excitation signals is adopted in IS54 as well as GSM6.20, while the algebraic structure in the fixed codebook is applied in ACELP, which is adopted in G.723.1 and G.729.

In this thesis, we propose the downsample method and modified codebook to cut down the computation in the close-loop search based on FS1016 standard. However, this method would suffer a little degradation in speech quality. Thus, the re-search and classifying method has adopted as a remedy. Finally, the proposed method can reduce the computational load drastically, but the speech quality is nearly the same with standard FS1016.



In the Chapter 2, the basic CELP structure and the standard FS1016 is introduced. In the Chapter 3, the proposed downsample method which is aim at reducing the computational load is introduced, and the modified stochastic codebook is proposed. Besides, the re-search and classification method which increase the speech quality is added too. In the Chapter 4, the simulation and the subjective testing are shown. Finally, the conclusion and future work are in the Chapter 5.

Chapter 2

Code Excited Linear Prediction (CELP) and FS1016

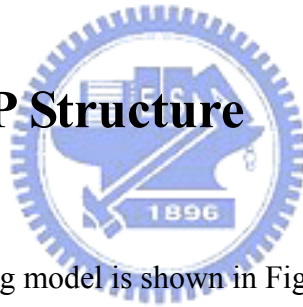


To encode the speech signals to low bit-rates, CELP is a suitable method which was cited by Atal in 1985 [1]. CELP adopt the method of analysis-and-synthesis (ABS) and use classical LPC method to encode the speech. By this way, CELP can simulate the process of vector quantization and use the source-filtered model at the same time. Differing from classical vocoders which use random noise or periodical pulse as the excitation sources, CELP use white-noise-like codebook to simulate the whitening source and use long-term filter that incorporate with codebook to

simulate the pitch-like part of signals.

The Federal Standard 1016 (FS1016) of USA that we use in this thesis is a typical example of 4.8 kb/s CELP coders. Having some clever modifications from basic principle of CELP, the FS1016 improve the computational efficiency and quality. Although there are many modern standards outperform the FS1016, the design of the FS1016 is a milestone in the speech coding development.

2.1 Basic CELP Structure



The basic CELP encoding model is shown in Fig 2-1.

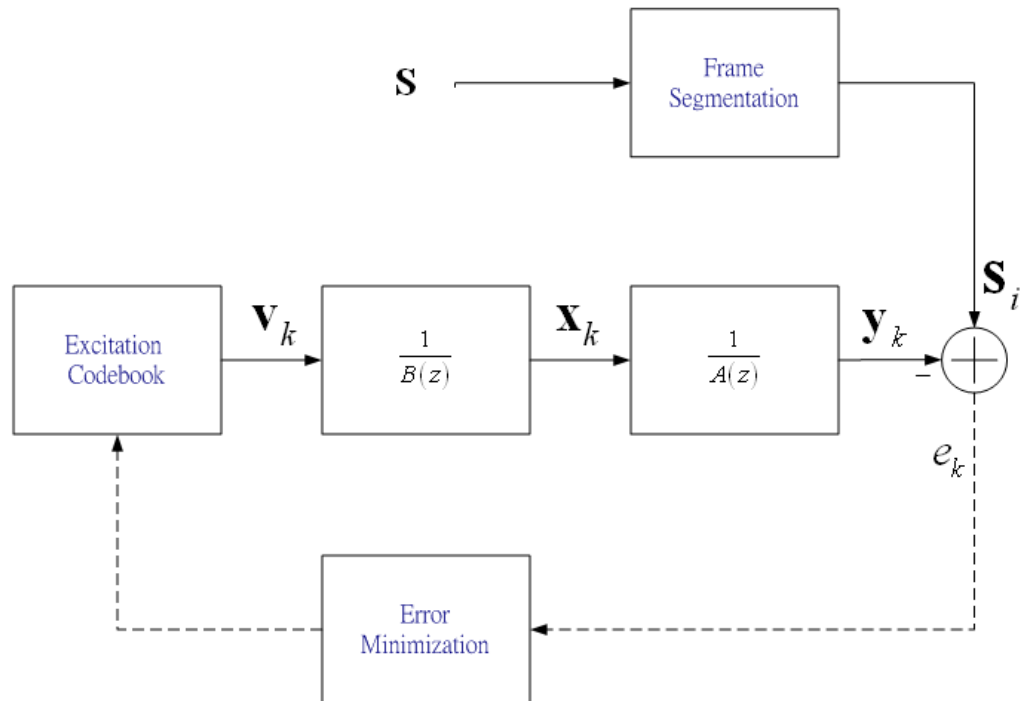


Figure 2-1 Basic CELP structure

In this model, speech is segmented to \mathbf{s}_i with length L , and the CELP also generates a vector \mathbf{y}_k which is like \mathbf{s}_i as much as possible. There are three main functional blocks in this model; they are excitation codebook, long-term filter and short-term filter. The excitation codebook is composed of N white-noise-like vectors and we denoted each vector as \mathbf{v}_k , where $k = 1, 2, \dots, N$. To generate a synthesized speech which resemble the original speech as much as possible, this model filter the excitation codebook vectors \mathbf{v}_k through long-term filter $\frac{1}{B(z)}$ and short-term filter $\frac{1}{A(z)}$ where the long-term and short-term filters are determined by analyzing \mathbf{s}_i .

The short-term filter is an all pole system, and its coefficients are so called LPCs (linear prediction coefficients) which can be obtained by using the classical method LP (linear prediction) analysis in s_i . By doing this, we can easily get LPCs to form this short-term filter. In the long-term filter, the pitch of s_i must be determined first and the coefficients in $A(z)$ is calculated and determined in turn.


After these three function blocks, the synthesis speech candidate y_k is obtained. The y_k , which is made from one of the codebook vectors v_k , must compare with the original speech segment s_i , and in turn, we can obtain the error e_k between v_k and s_i , that is usually obtained by using square error method. So, each v_k maps to one e_k , and by sorting e_k , where $k = 1, 2, \dots, N$, we can determine the “best” excitation vector v_{best} that produce the smallest error e_{best} , where $e_{best} < e_k$, where $k = 1, 2, \dots, N$ and $k \neq best$. This also means that we can produce y_{best} , which is like s_i most, by filtering v_{best} through long-term and short-term filter. Note the process that we choose the v_{best} by using the synthesized speech y_k and this procedure is so called ABS (analysis-by-synthesis).

Hence, in the decoding end, we only need to know the codebook index, long-term filter and short-term filter coefficients, so that we can produce the same y_{best} , that has been generated in the encoding process.

2.2 The FS1016 Standard

The FS1016 standard is based on CELP model and has some modification of CELP's basic structure to meet some requirements of implementation. Further more, FS1016 adds detailed adjustments which is based on experimented results to gain some quality. In this Section, a basic modified structure, some main functional blocks, and specifications of this standard will be introduced.

2.2.1 Overview of Standard



FS1016 is a speech compression standard and the goal of compression standard is to encode the original speech to a shortest bit stream and still have a reasonable speech quality after decoding. FS1016 takes a speech which is sampled at 8k/sec with 16 bits resolution of each sample and encodes the speech to a bit stream at the bit-rate of 4800 bits per second. Thus, FS1016 achieves the compression ratio to 26.67, yet the quality is acceptable.

First, the speech is segmented into frames, and each frame has the length of 30ms; that is, each frame consists of 240 samples of the speech. Then every frame will go through some processes to obtain and quantize LPCs. Here, ten order LPCs is

adopted to achieve the balance between computation and preciseness based on experimental results. Afterward, because of the preciseness of pitch, each frame will be further divided into four shorter subframes, so that each subframe is composed of 60 speech samples and has duration of 7.5ms. Then every subframe will execute adaptive codebook search and stochastic codebook search. We should note that the codebooks here use the gain-shape codebook structure, which means that each codebook is concatenated with a multiplier which serves as the gain of the codevector.

Finally, all the parameters will be quantized to bits according to their own bit allocation and all bits will pack into a bit stream per frame. Then, the bit stream will transmit to the decoding end.

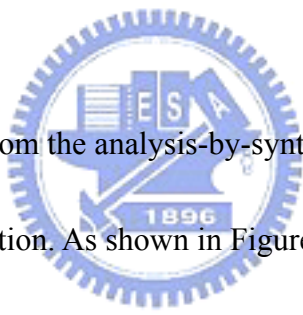
The bit allocation in FS1016 is summarized in Table 2-1.

Table 2-1 Bit allocation in FS1016 encoder.

Parameter	Bits per Frame
LPCs	34
Adaptive codebook index	$8+6+8+6=28$
Adaptive codebook gain	$5*4=20$
Stochastic codebook index	$9*4=36$
Stochastic codebook gain	$5*4=20$
Synchronization bit	1
Error correction bit	$1*4=4$
Future expansion bit	1
Total	144

In Table 2-2, the information (index and gain) are given per subframe in adaptive codebook and stochastic codebook, and so is error correction. That is why we should give four times bit allocations in the packing stage that performs once per frame. From this bit allocation table, we can know that there are $2^9 = 512$ vectors in the stochastic codebook. Besides, there are $2^8 = 256$ or $2^7 = 128$ pitches in the standard and some of these values are fractional.

2.2.2 The FS1016 structure



The FS1016 is derived from the analysis-by-synthesis structure of CELP that has mentioned in the last section. As shown in Figure 2-1, we can know that the analysis procedure is frame-by-frame basis. Because of this frame-by-frame feature, there is redundancy in the close-loop search. Thus, by using some DSP techniques, the redundancy in the close-loop search will be eliminated, and luckily, some other benefits will be obtained by the way.

In the signal processing, the filter response can be separated to zero-input and zero-state response. Applying this technique to short-term filter, we can get a new structure as shown in Figure 2-2.

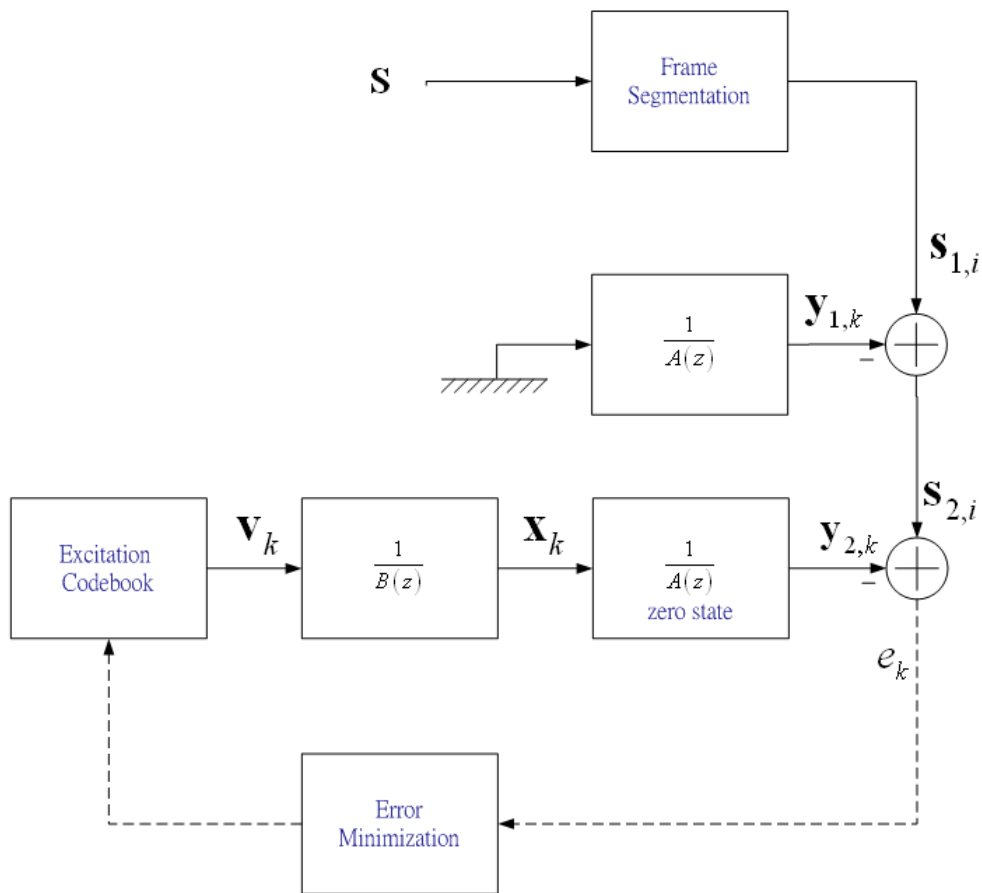


Figure 2-2 Separate the short-term filter response to zero-input response and zero-state response.

From Figure 2-2, we can see that the short-term filter is separated to zero-input filter and zero-state filter. Note that by doing this, the zero-input filter is outside the close-loop search. So, in the synthesis procedure, we do not need to calculate the contribution of zero-input response from x_k to $y_{2,k}$ in the beginning and this will save $10+9+\dots+1=55$ multiplications per iteration in the loop. Therefore, we can

totally save 55×512 multiplications in the close-loop search. Further more, the standard specifies the close-loop search in the subframe basis rather than frame basis; that is, when every subframe doing the short term filtering once, we can save 55 multiplications. Thus, $4 \times 55 \times 512$ multiplications are saved per frame in the close-loop search.

In FS1016, this technique is applied again on the long-term filter and the structure is shown in Figure 2-3.



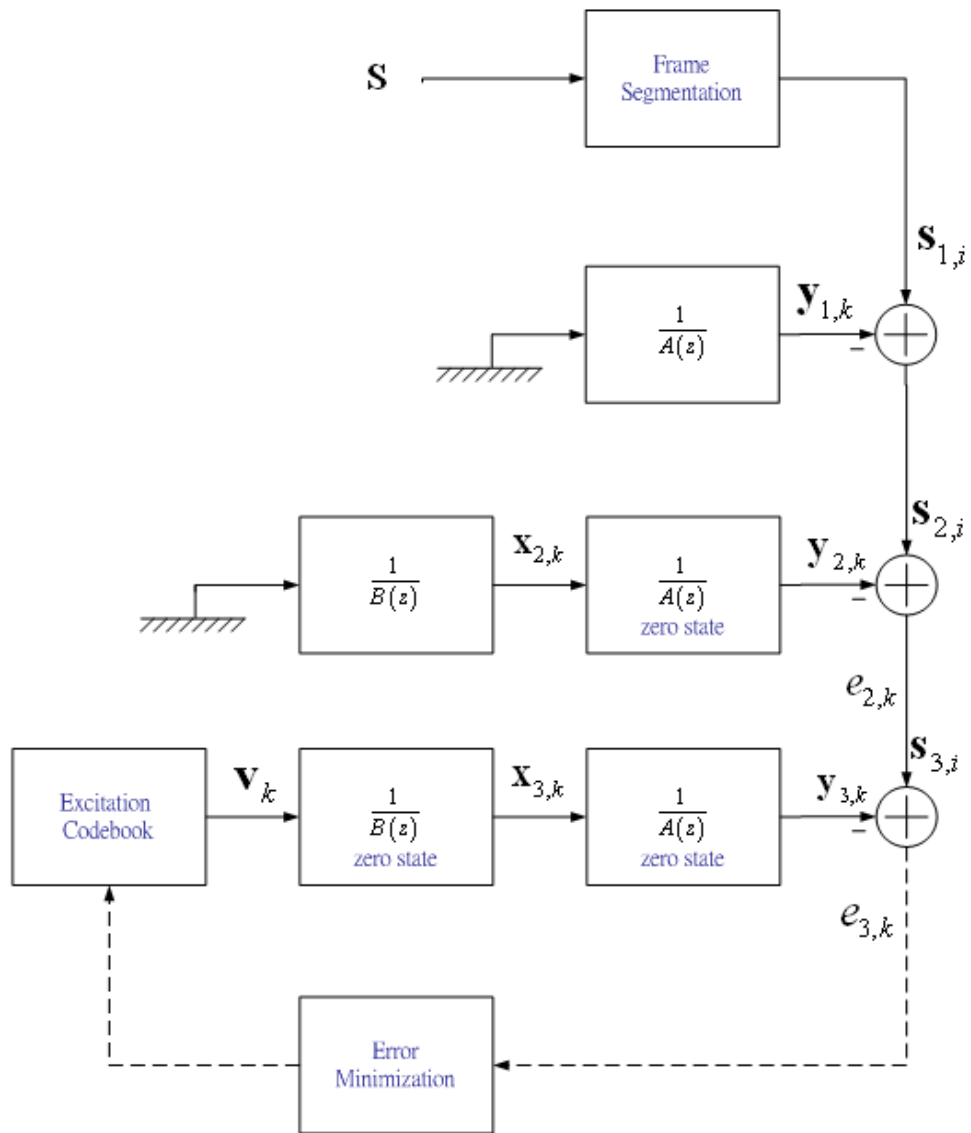


Figure 2-3 Separate the long-term filter to zero-input response and zero-state response further.

Note that the $y_{2,k}$ here is different from that in the Figure 2-2. Again, using the technique that separates the filter response to zero-input and zero-state response, the zero-input response of the long-term filter can be taken outside the codebook search loop. Yet the purpose here is not only saving multiplications as mentioned above,

but also incorporating an adaptive codebook into this structure to get a new way to estimate the pitch efficiently. That will discuss later. So, the structure of the FS1016 is shown in Figure 2-4.

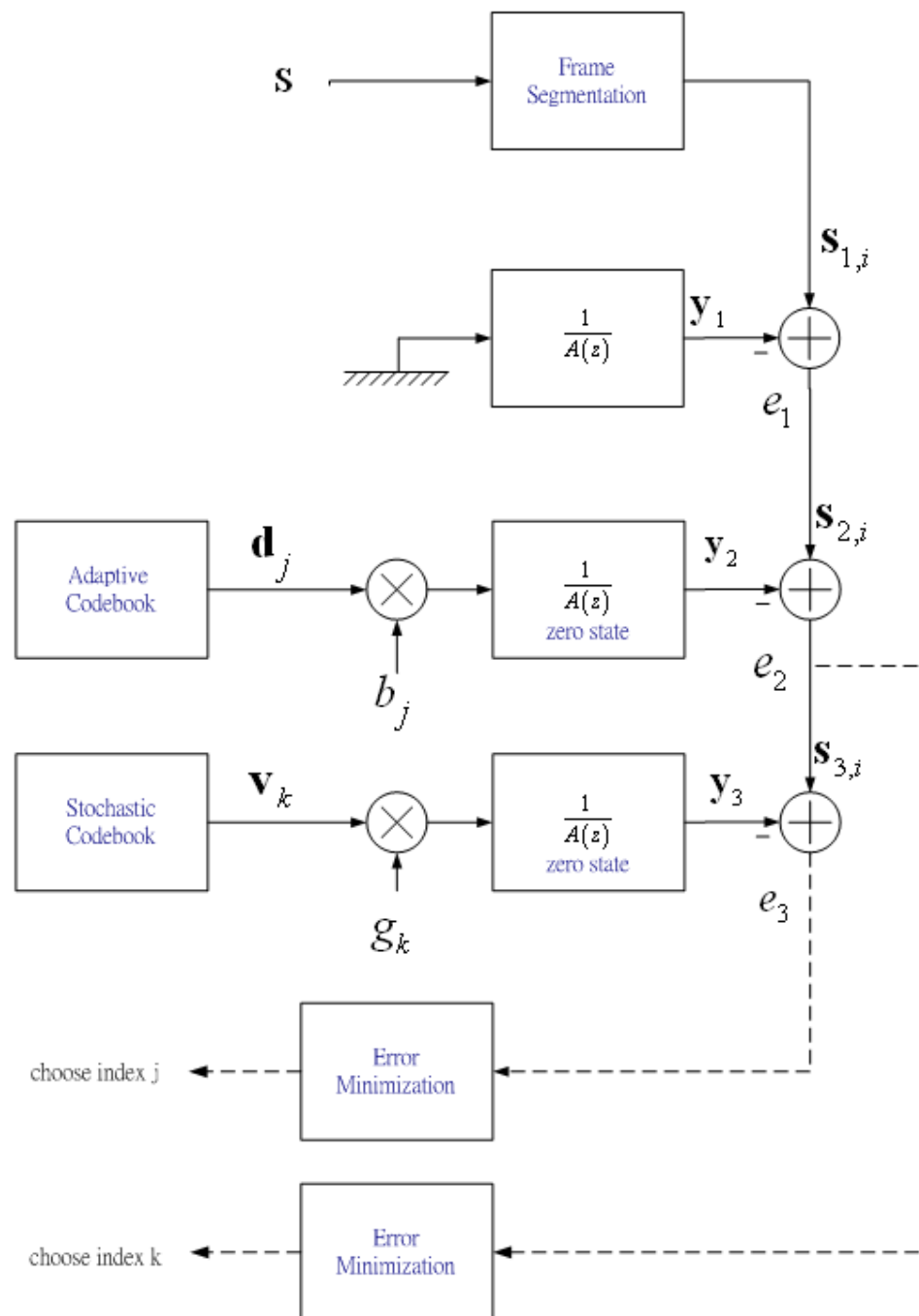


Figure 2-4 The structure in FS1016.

Here, the excitation codebook is renamed to stochastic book for emphasizing its white-noise-like characteristics, and the zero-input long-term filter is replaced to an adaptive codebook. And by incorporating the adaptive codebook, another close-loop in the structure is formed. Note that adaptive codebook and stochastic codebook both use so-called gain-shape codebook structure, so each codebook must be concatenated with a multiplier to serve as the gain of the codebook. Someone may notice that the long-term filter is eliminated. That is because the adaptive codebook, which is derived from zero-input long-term filter, is responsible for the periodic part of the input speech, and the stochastic codebook is responsible for the “random” part. Thus, we can eliminate the long-term filter behind the stochastic codebook, and there is little difference between the taken one and the original one through subject tests. In turn, by taking the zero-state long-term filter away, the computation is further reduced.

So far in this section, the structure of encoder is introduced, so the structure of the decoder does not differ much with encoder in the reverse process. The decoder structure is shown in Figure 2-5.

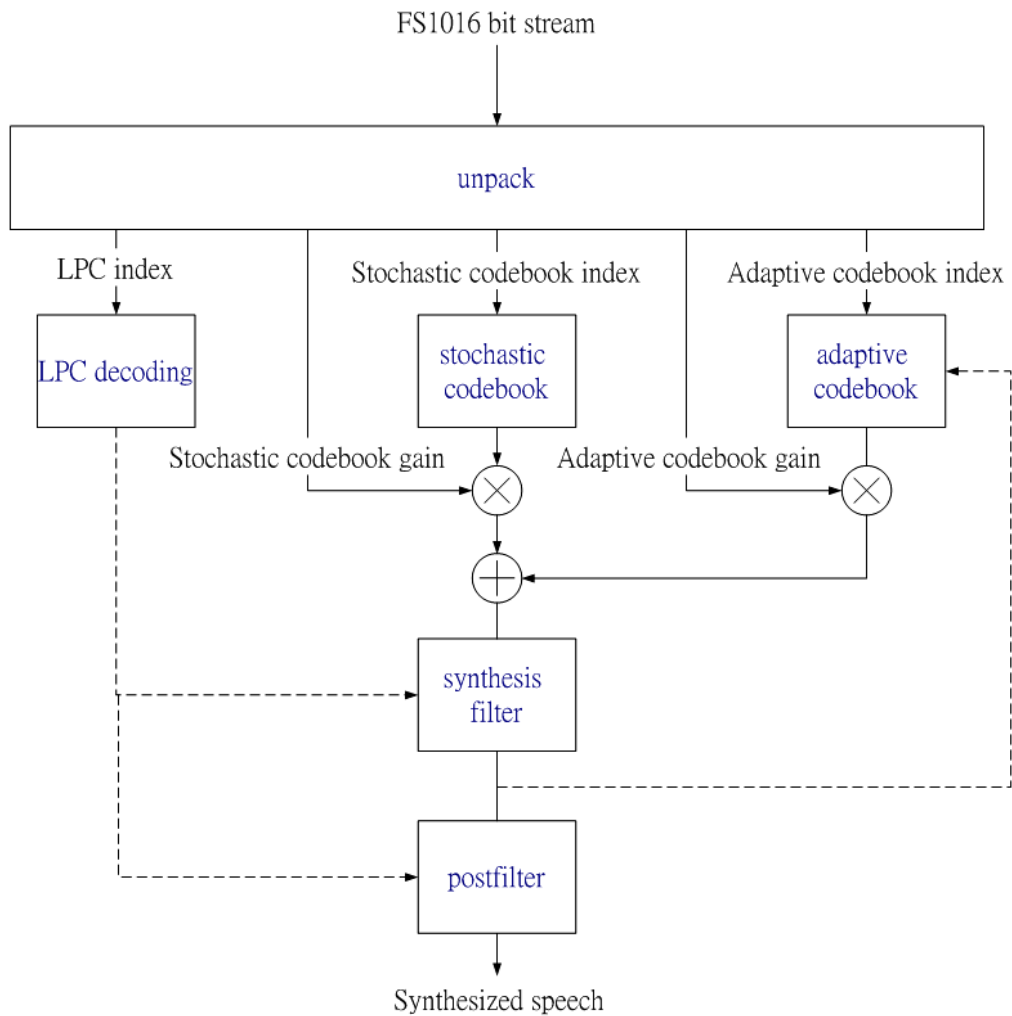


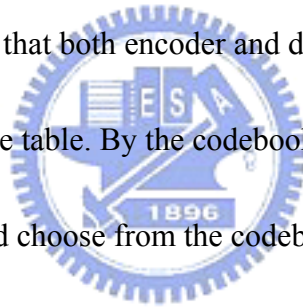
Figure 2-5 The structure of FS1016 decoder.

In the decoder, the first step is unpacking the bit-stream and value of LPC index, so codebook index and gain in both adaptive codebook and stochastic codebook are known. In turn, the corresponding codebook vectors in both stochastic codebook and adaptive codebook are extracted. Besides, the LPC is also gotten, and the synthesis filter and postfilter can be formed from LPC. Then, the two codebook vectors add together and filtered by the synthesis filter and the postfilter. Finally, the

synthetic speech is formed. Note that the synthesis filter here is a replicate of short-term filter. However, the filter here is not separated to zero-input part and zero-state part, because there is no codebook search issue here.

2.2.3 Adaptive Codebook

There are two codebooks in the FS1016: adaptive codebook and stochastic codebook. Codebook means that both encoder and decoder have the same table, and decoder use the vectors in the table. By the codebook index encoder sends, decoder can know what vector should choose from the codebook.



The term “adaptive” here means that this codebook changes. As mentioned above, the adaptive codebook is evolved from the long-term filter $\frac{1}{B(z)}$, so the contents of the adaptive codebook are period-like signals. Now, we introduce how long-term filter evolve into adaptive codebook.

The transfer function of the long-term filter is

$$\frac{1}{B(z)} = \frac{1}{1-bz^{-T}} \quad (2.1)$$

There are two undetermined parameter b and T here. To get them, we can adopt close-loop search to find the optimal values. This is illustrated in Figure 2-6.

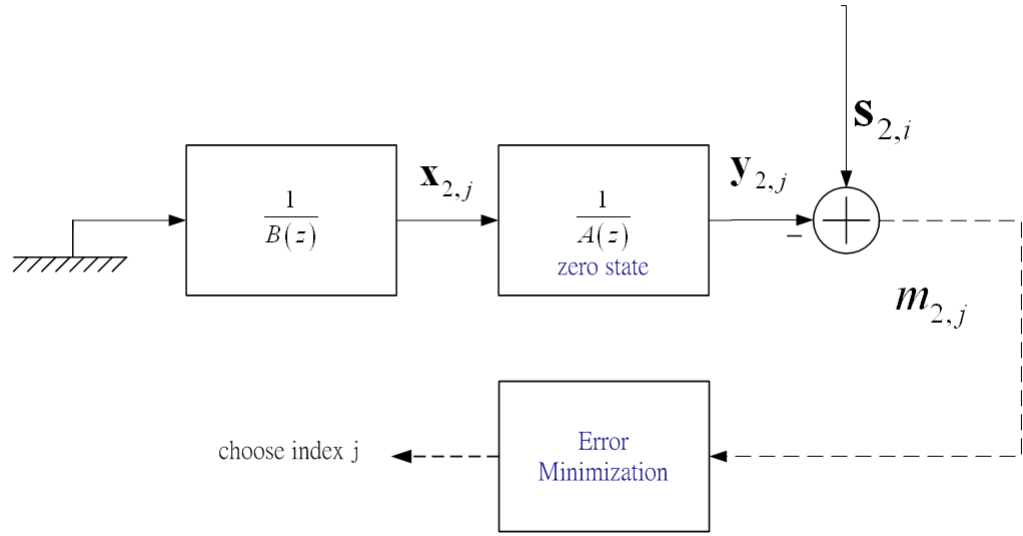


Figure 2-6 Close-loop search structure with zero-input long-term filter.

Here, the $x_{2,j}$ is the zero-input response the long-term filter. After $x_{2,j}$ is gotten, the $y_{2,j}$ is produced and the likeness between $s_{2,i}$ and $y_{2,j}$ can be measured by $m_{2,j}$. The $m_{2,j}$ is produced by

$$m_{2,j} = \frac{\left(\sum_{n=0}^{L-1} y_{2,j}[n] s_{2,i}[n] \right)^2}{\sum_{n=0}^{L-1} y_{2,j}[n] y_{2,j}[n]}, \quad j = 0, 1, 2, \dots, N_a - 1 \quad (2.2)$$

where N_a is the number of combination results with parameters b and T .

Through $m_{2,j}$, the maximum $m_{2,j}$ from $j = 0, 1, 2, \dots, N_a - 1$ can be found, and the corresponding index j is known. Thus, the corresponding parameters b and T can be known.

In the FS1016 standard, the zero-input response is simplified to replicate the

synthesized signals [1]. By doing this, the complexity is reduced, and the parameters b and T are transformed to codebook gain and codebook. The structure is shown below:

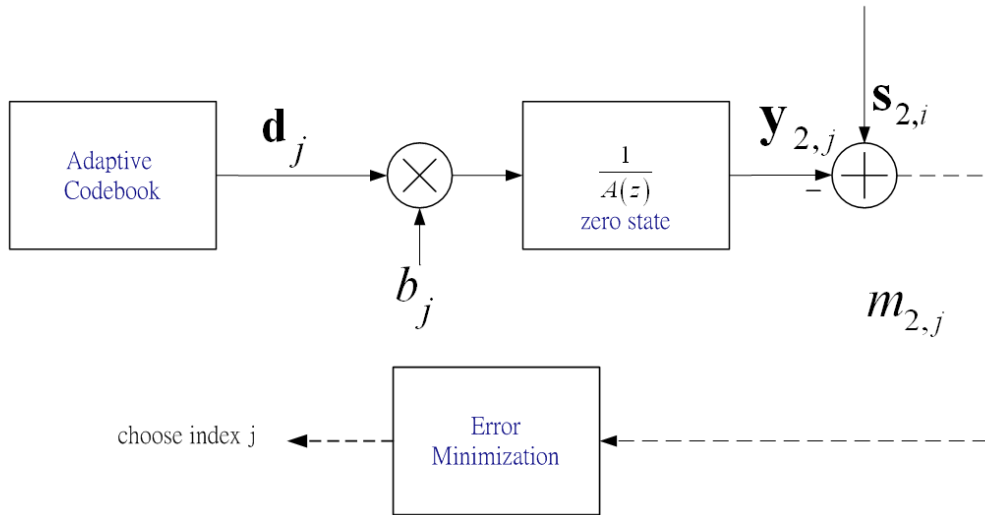


Figure 2-7 The adaptive codebook search structure.

Here, the codebook gain b_j is obtained by

$$b_j = \frac{\mathbf{s}_{2,j}^T \cdot \mathbf{y}_{2,j}}{\|\mathbf{y}_{2,j}\|^2} \quad (2.3)$$

and $m_{2,j}$ is produced by

$$m_{2,j} = \frac{(\mathbf{s}_{2,j}^T \cdot \mathbf{y}_{2,j})^2}{\|\mathbf{y}_{2,j}\|^2} \quad (2.4)$$

Then, by finding the maximum $m_{2,j}$, the codebook index j is determined, so the corresponding codebook gain is also determined.

The most important advantage of this structure is that the codebook size is related to the number of possible values of parameter T rather than the number of combination of parameter b and T . Thus, the codebook size is reduced a lot, but the quality is nearly the same.

In the standard, there are totally 256 values for the parameter T , ranging from $T = 20$ to $T = 147$, including some fractional values in some specific intervals.

2.2.4 Stochastic codebook

The term “stochastic” means that this codebook contains random vectors. In the standard, the zero-mean unit-variance white Gaussian source is adopted.

The stochastic codebook contains N codevectors with length of L where $N = 512$ and $L = 60$ are adopted in the standard, and the stochastic codebook is stored in the memory in the both decoder and encoder. However, rather than storing $N \times L = 512 \times 60 = 30720$ values in the memory, the standard applies overlapping structure. With shift $s = 2$, there are $L - s$ values in common between the neighboring vectors, there are just $s(N - 1) + L = 1028$ values that have to save in the memory. Nevertheless, in the overlapping structure, there is no difference with non-overlapping structure in subjective testing, but the gain the profit of reducing

the required memory. The overlapping structure is illustrated in the Figure 2-8.

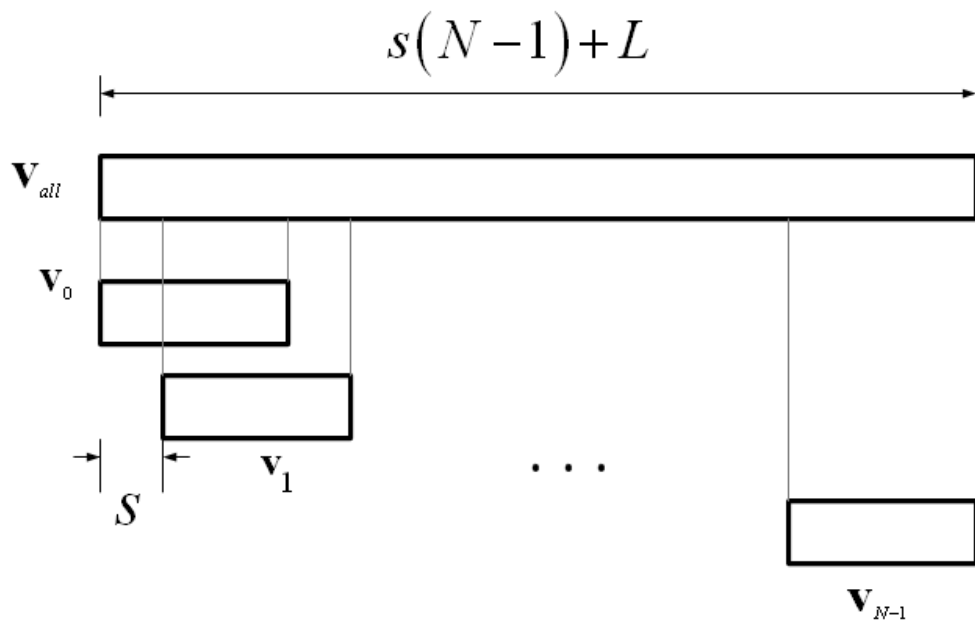


Figure 2-8 Overlapping structure in stochastic codebook.

In the standard, the 1082 values in stochastic codebook are provided by the National Communications System. In the stochastic codebook, every codebook vector is a zero-mean unit-variance Gaussian white signal. However, the values in stochastic is specially designed, so it is not just a random produced random signal. In the codebook, there are about 77% values are zeros. Moreover, the codebook contents are just tree specific values: 1, 0, -1. Due to these two features of codebook, the computational load can be cut down drastically; by the way, required memory is

reduced too. However, the quality is not sacrificed by getting these benefits.

The stochastic codebook also incorporates with synthesis filter to form a close-loop. Like the adaptive codebook, the stochastic codebook is a gain-shape codebook, so the codebook must connect with a multiplier as a gain. The structure is shown below.

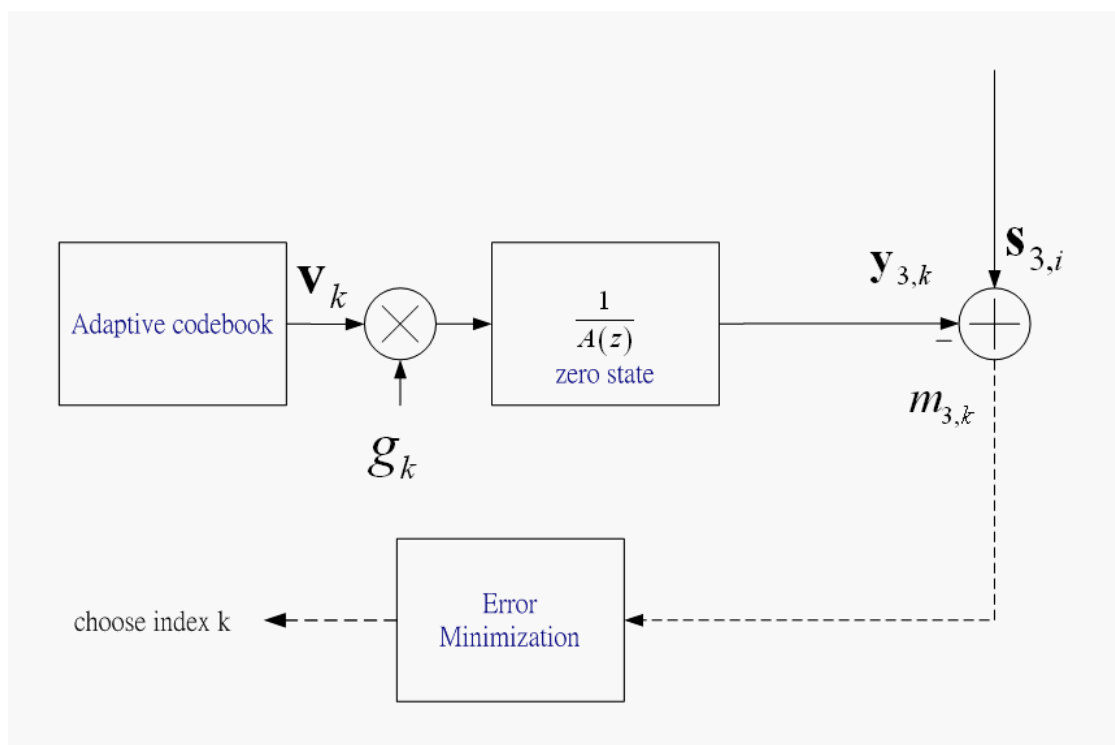


Figure 2-9 The close-loop structure of stochastic codebook.

The codebook gain g_k here is obtained by

$$g_k = \frac{\mathbf{s}_{3,k}^T \cdot \mathbf{y}_{3,k}}{\|\mathbf{y}_{3,k}\|^2} \quad (2.5)$$

and $m_{3,k}$ is produced by

$$m_{3,k} = \frac{(\mathbf{s}_{3,k}^T \cdot \mathbf{y}_{3,k})^2}{\|\mathbf{y}_{3,k}\|^2} \quad (2.6)$$

Then, by finding the maximum $m_{3,k}$, the codebook index k is determined, so the corresponding codebook gain is also determined.



Chapter 3

Downsample Codebook Search with Modified Codebook



In the Chapter 2, the basic concepts of FS1016 have been introduced; although, the complexity of FS1016 is much lower than direct implementation by CELP, the computation load, however, is heavy as well. In the FS1016 structure, most computation load lies in codebook search (both adaptive codebook search and stochastic codebook search), so to ease the codebook-searching load is an important issue in applications.

The codebook search in FS1016 is a close-loop search, and we can separate this

procedure into three parts. The first part is synthesis that one of the codebook vectors will be filtered and generates a candidate vectors. The second part is match calculation that the candidate vectors will compare to the target vectors and produce a score in turn. And the third part is choosing the vector that has maximum match according to the scores generated in second part.

In the stochastic codebook search, if the codebook size is determined, the computation load of the third part is determined too. In this thesis, we don't intend to change the codebook size, so reducing the codebook size is not our goal.

However, the synthesis and match calculation procedure can be reduced.

In this chapter, we propose a downsampled codebook search, which can cut down the codebook-searching computation directly in the first and second part (synthesis and match calculation), and we can further reduce the stochastic codebook-searching computation load in the first part (synthesis) by incorporating the modified codebook. However, the quality using reduced-points codebook search is sacrificed a little. Thus, the re-search and classified method will be incorporated to enhance the quality but induce some complexity. However, the totally computation load is much lower.

3.1 Downsample in Match Calculation

This section will focus on the computational reduction in the second part (match calculation).

In the Chapter 2, we know that the stochastic codebook uses gain-shape codebook structure, so we can obtain the match between $\mathbf{y}_{3,k}$ and $\mathbf{s}_{3,i}$ using correlation between $\mathbf{y}_{3,k}$ and $\mathbf{s}_{3,i}$:

$$m_k = \frac{(\mathbf{y}_{3,k}^T \cdot \mathbf{s}_{3,k})^2}{\|\mathbf{y}_{3,k}\|^2}, \quad k = 0, 1, 2, \dots, N-1 \quad (3.1)$$

To get m_k , we need $2L+2$ multiplications, so $N(2L+2)$ multiplications is needed within the one close-loop search. Instinctively, the number of multiplication can be cut down directly, if we downsample $\mathbf{y}_{3,k}$ and $\mathbf{s}_{3,i}$ to a shorter length l ; that is, we change $\mathbf{y}_{3,k}$ to another vector $\mathbf{y}_{3,k}^{(l)}$, and replace $\mathbf{s}_{3,k}$ with $\mathbf{s}_{3,k}^{(l)}$. Here, we can obtain the $m_k^{(l)}$ using Equation (3.1) by changing the $\mathbf{y}_{3,k}$ and $\mathbf{s}_{3,i}$ to $\mathbf{y}_{3,k}^{(l)}$ and $\mathbf{s}_{3,i}^{(l)}$. This is shown in Equation (3.2).

$$m_k^{(l)} = \frac{(\mathbf{y}_{3,k}^{(l)T} \cdot \mathbf{s}_{3,k}^{(l)})^2}{\|\mathbf{y}_{3,k}^{(l)}\|^2}, \quad k = 0, 1, 2, \dots, N-1 \quad (3.2)$$

Through Equation (3.2), we can get N $m_k^{(l)}$ per close-loop search. After N $m_k^{(l)}$ are obtained, we also choose the codebook vector that has maximum $m_k^{(l)}$.

However, the reduction of the vector length will lead to producing a different set of

m_k that we denote it as $m_k^{(l)}$, and in turn result in choosing the different codebook index. In this thesis, we assume that we can get the best codebook index with L vector length, where $L > l$. Thus, we call that we get the wrong codebook index if we get the different codebook index in length L and l . Because changing $\mathbf{y}_{3,k}$ to $\mathbf{y}_{3,k}^{(l)}$ is a linear transform $\mathbf{y}_{3,k}^{(l)} = \mathbf{P}\mathbf{y}_{3,k}$ that project one vector from \mathbb{R}^L to its subspace \mathbb{R}^l where \mathbf{P} is a projection matrix, we can not be sure we can always choose the right one after transformation.

To illustrate this, we assume $\mathbf{y}_k^{(3)}$ and $\mathbf{s}^{(3)}$ are vectors in \mathbb{R}^3 , and all vectors are transformed to a 2-dimensional space \mathbb{R}^2 through projection matrix $\mathbf{P}_{2 \times 3}$ as illustrated below.



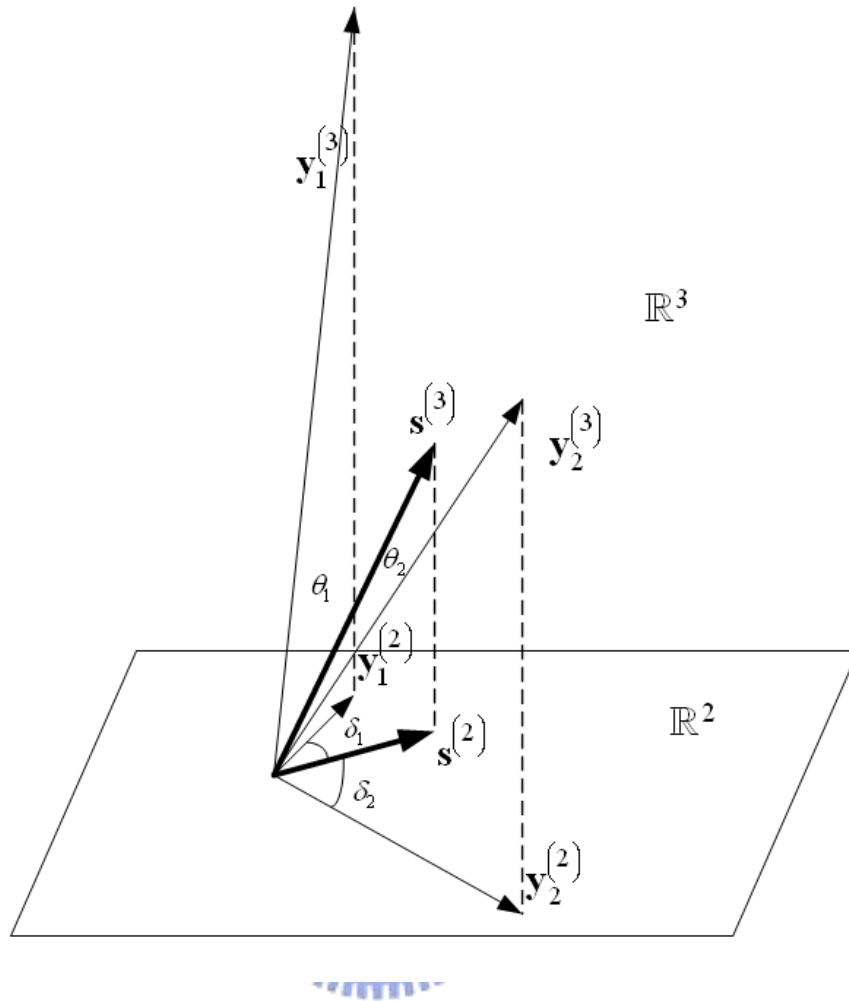


Figure 3-1 The space diagram.

As shown in Figure 3-1, in the 3-dimensional space, $\mathbf{y}_2^{(3)}$ will be chosen because the θ_1 , the angle between $\mathbf{s}^{(3)}$ and $\mathbf{y}_1^{(3)}$, is smaller than θ_2 , which is the angle between $\mathbf{s}^{(3)}$ and $\mathbf{y}_2^{(3)}$. However, in the 2-dimensional space, the situation may be different. In the Figure 3-1, the transformed vector $\mathbf{y}_1^{(2)}$ is chosen instead due to $\delta_1 < \delta_2$. Therefore, we can reduce the computational load by comparing all the candidate vectors in the subspace but take the risk of choosing the wrong codebook index.

Here, we refer the term “hit” as choosing the right codebook index in the subspace \mathbb{R}^l , and refer “miss” as choosing the wrong codebook index. Through the probability of hit and probability of miss, we can evaluate the risk of choosing the wrong index; that is, if we are in the case that the probability of hit is high, we are sure to achieve the computational reduction by sacrificing less quality.

Here, some experiment will show. We apply this downsample codebook search to the stochastic codebook in our speech database. Below, all subframes of speech, which have 60 points, are transfer to the different subspace \mathbb{R}^l , where $l = 10, 20, 30, 40, 50, 60$, and calculate its probability of hit.

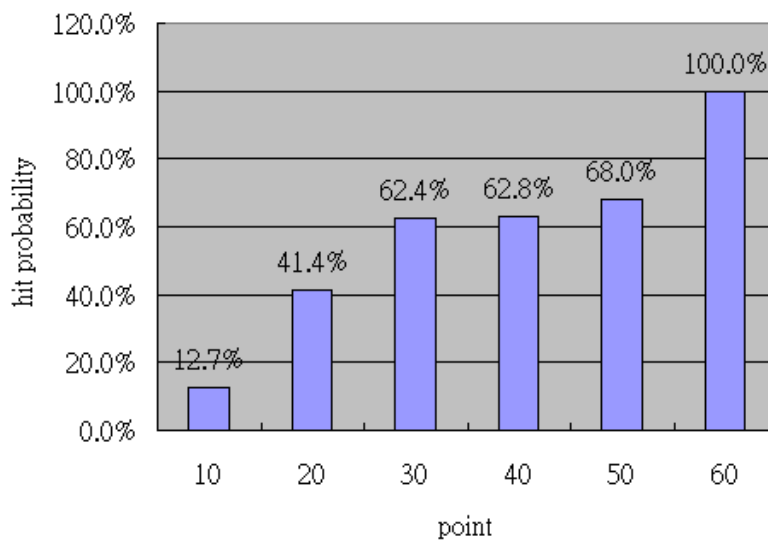


Figure 3-2 The probability of hit in \mathbb{R}^l , where $l = 10, 20, 30, 40, 50, 60$.

As shown in Figure 3-2, the probability of hit decreases in approximately linear

scale. Table 3-1 is the comparison between the computational load in match calculation and probability of hit, that we pick in $l = 10, 20, 30, 40, 50, 60$.

Table 3-1 The probability of hit in $l = 10, 20, 30, 40, 50, 60$.

Subspace dimension l	Computation in Match	The probability of hit
10	16.66%	12.7 %
20	33.33%	41.4 %
30	50%	62.4 %
40	66.66%	62.8 %
50	83.33%	68.0 %
60	100%	100.0 %

As we can see in Table 3-1, we reduce the computational load at the price of less quality decrease. For example, when subspace dimension is 30, which means we reduce the computational load to half, but the probability of hit is higher than $\frac{1}{2}$.

3.2 Modified Codebook

In this section, we reduce the computation load in the synthesis part by downsample. Moreover, the modification codebook will induce in turn to cut down the computation further based on the downsample method, yet the quality will not

be further sacrificed.

As mentioned above, there are three parts in the close-loop search: synthesis, match calculation, and choosing the maximum match. In section 3.1, the match calculation complexity is reduced by reduce the reduce-points manner at the price of poorer quality. In this section, we will reduce the complexity of synthesis part in the close-loop search, but not affect the quality.

In the Section 3.1, we transfer the signal to subspace by specific patterns. Some patterns serve the function as downsample. So, when system operating in these situations, the close-loop search can be illustrated in Figure 3-3.

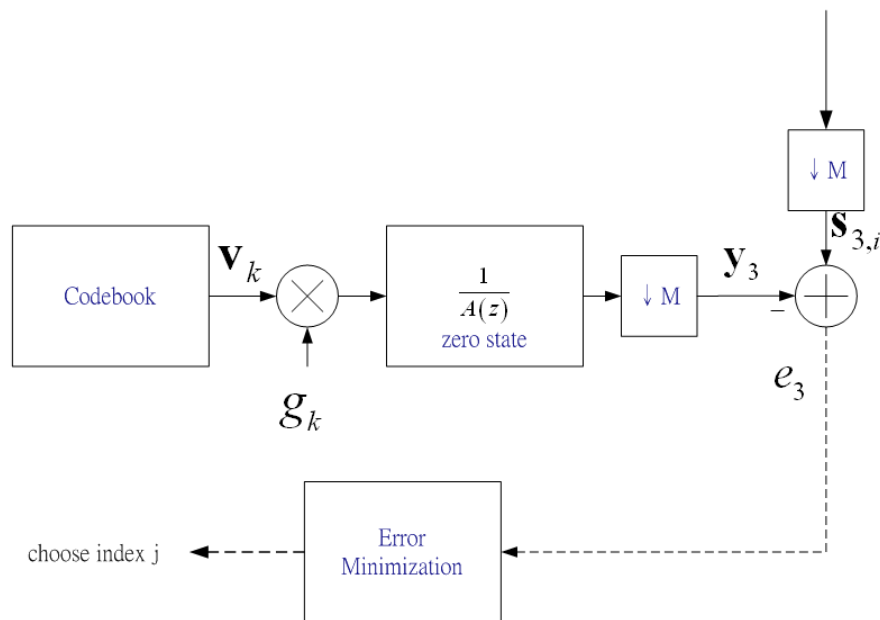


Figure 3-3. The close-loop search incorporates with downsampling search.

As shown in Figure 3-3, we downsample the $y_{3,k}$ and s_{3i} by M respectively before match calculation. Moreover, we can move the downsampler from the output-end of the filter $\frac{1}{A(z)}$ to the front-end to reduce the filtering calculation to $\frac{1}{M}$ times.

As mentioned in Chapter 2, the short-term filter $\frac{1}{A(z)}$ is a IIR filter with order of 10. To move the downsampler through the filter $\frac{1}{A(z)}$, we have to do the polyphase decomposition as followed:

$$\frac{1}{A(z)} = H_0(z^M) + z^{-1}H_1(z^M) + \dots + z^{-M+1}H_{M-1}(z^M) \quad (3.3)$$

Then, we can move the downsampler through $H_f(z^M)$ as below:

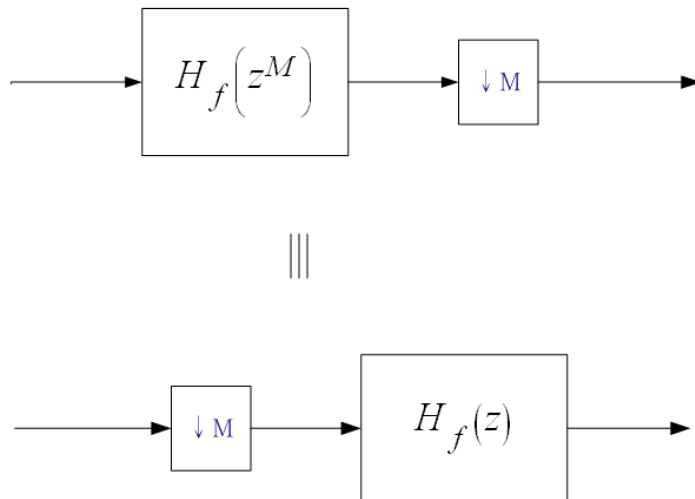


Figure 3-4 Exchange the filter and downsampler.

After moving the downsampler to the front of filter, it is easy to see that we can reduce the multiplication and addition to about $\frac{1}{M}$. However, as shown in Equation (3.5), it is hard and computation intensive to get the $H_f(z^M)$, where $f = 0, 1, 2, \dots, M - 1$. Moreover, $H_f(z^M)$ may be infinite impulse response as $\frac{1}{A(z)}$, so it is not possible to apply this to real applications.

Here, there is a good way to approximate the polyphase decomposition of $\frac{1}{A(z)}$. Firstly, after $\frac{1}{A(z)}$ is obtained through LPC, we can turn the filter $\frac{1}{A(z)}$ to FIR filter $H(z)$ by applying an impulse to the input of the filter. Note that the filter $\frac{1}{A(z)}$ is zero-state filter, so the response will not be affected by previous input; thus, we can share the same $H(z)$ in the four subframes within the same frame. This process of turning IIR filter to FIR filter [1] is illustrated as followed:

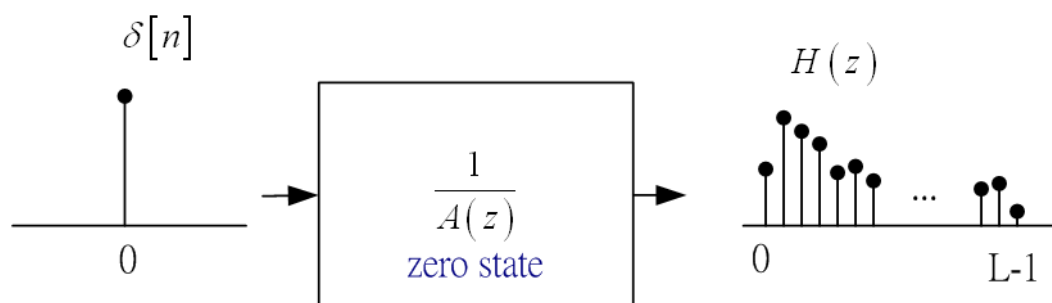
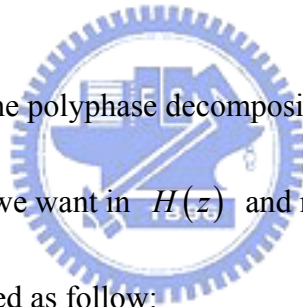


Figure 3-5 Turning IIR filter to FIR filter.

After giving an impulse signal to the input of the filter $\frac{1}{A(z)}$, the response $H(z)$ may be a have infinite impulse response. However, we do not need all of these. The response that exceeds the subframe length L have no contribution to $y_{3,k}$, so we only get $H(z)$ up to order $L-1$, as shown in Figure 3-5. Note that the stochastic codebook is a ternary-value; that is, there are only three values in the codebook: 1, 0, -1. So, we can take advantage of this feature and just use shift-and-add to complete the filtering without multipliers. Thus, the complexity can be further reduced.

Next, we can easily do the polyphase decomposition of $H(z)$ to get $H_f(z)$ by just picking the samples we want in $H(z)$ and move the downsampler through the $H_f(z)$. This is illustrated as follow:



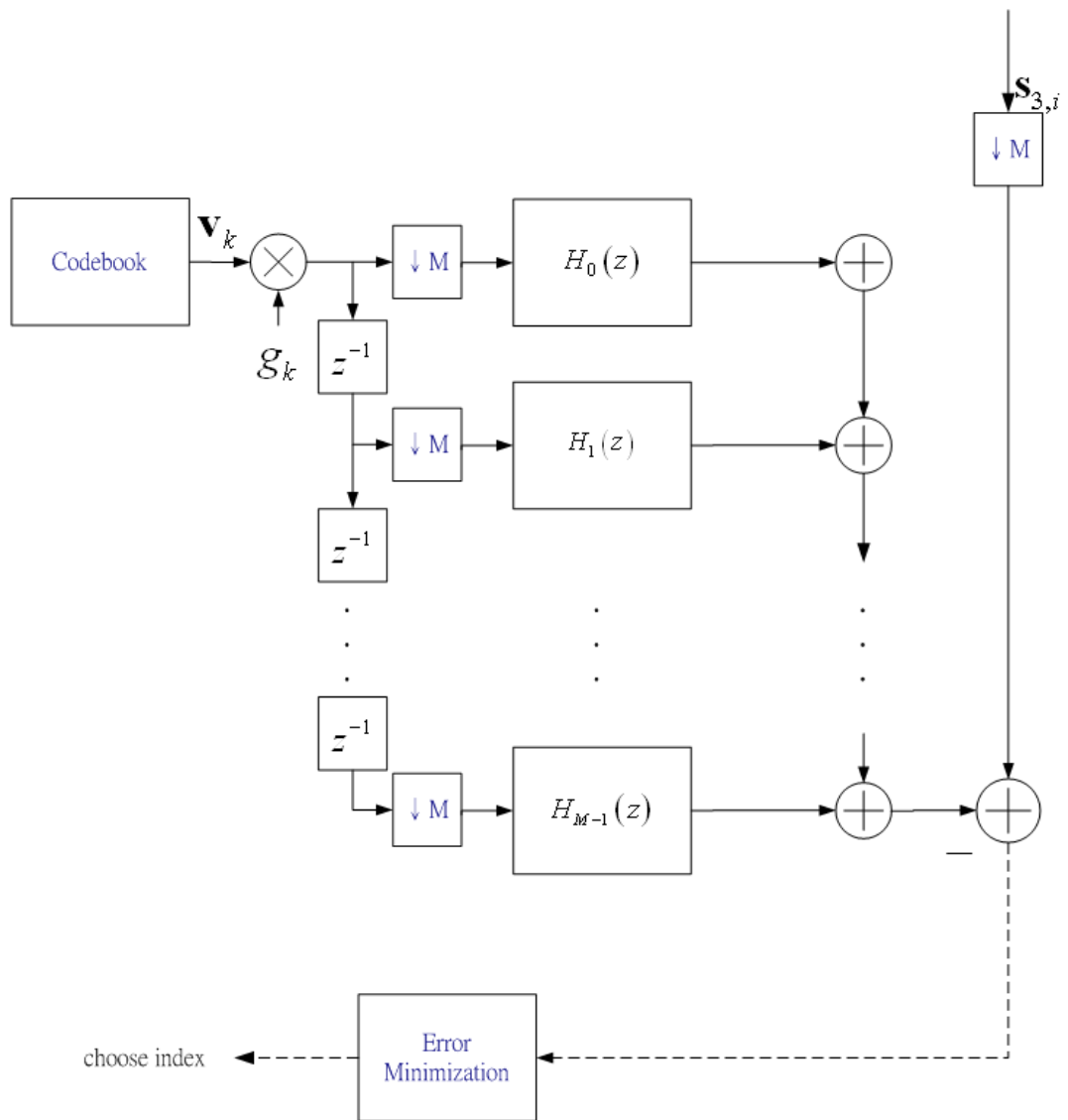


Figure 3-6 Moving downsample to the front-end of the filter.

The Figure 3-6 shows the structure after moving the downsampler to the front-end of the short-term filter $\frac{1}{A(z)}$. Note that the $\mathbf{y}_{3,k}^{(l)}$ will be the same after this rearrangement; this means that this rearrangement will cut down the computation but not affect the quality.

So far, we have introduced the downsample method that reduces the computational load of not only match calculation but synthesis filtering as well; moreover, we can use the this polyphase structure to further cut down the computation.

Here, we take downsample by two as an example to illustrate this idea and the basic structure is shown in Figure 3-7.

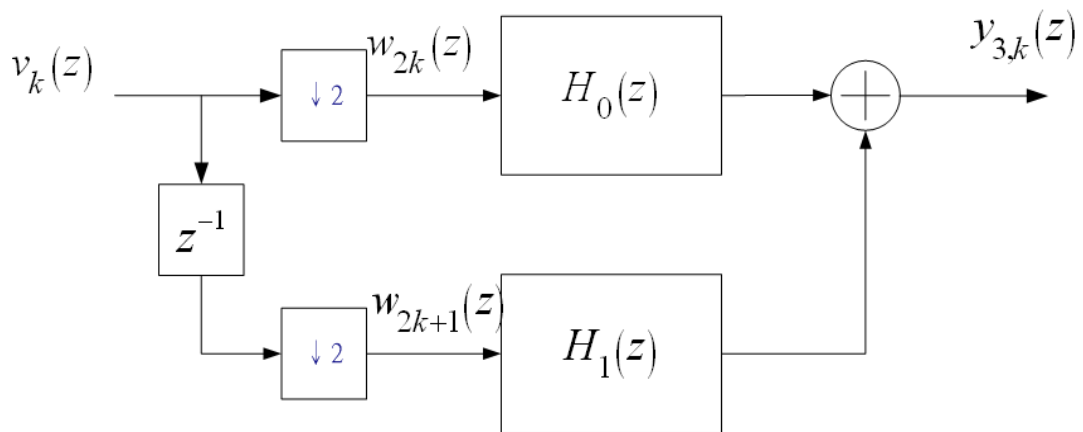


Figure 3-7 Polyphase structure with downsample two.

In Figure 3-7, we can see that

$$y_{3,k}(z) = w_{2k}(z)H_0(z) + w_{2k+1}(z)H_1(z) \quad (3.4)$$

There, the $w_{2k}(z)$ and $w_{2k+1}(z)$ are the polyphase components of codebook vector $v_k(z)$. So, we can know that

$$v_k(z) = w_{2k}(z^2) + z^{-1}w_{2k+1}(z^2) \quad (3.5)$$

In one close-loop search, the index k will iterate from 0 to $N-1$ every subframe, so there are N pairs of $w_{2k}(z)$ and $w_{2k+1}(z)$ will serve the inputs of polyphase filters $H_0(z)$ and $H_1(z)$ to generate N different candidate vectors $y_{3,k}(z)$. In other words, there are $2N$ different polyphase filters inputs $w_r(z)$, where $r = 0, 1, \dots, 2N-1$, to generate N different candidate vectors $y_{3,k}(z)$.

However, if we want to get N different $y_{3,k}(z)$ to represent N different phase information, we only need N different $w_r(z)$ rather than $2N$ different ones. This can be done by rearranging the placement of $H_f(z)$ and $w_r(z)$ as

shown in Figure 3-8.



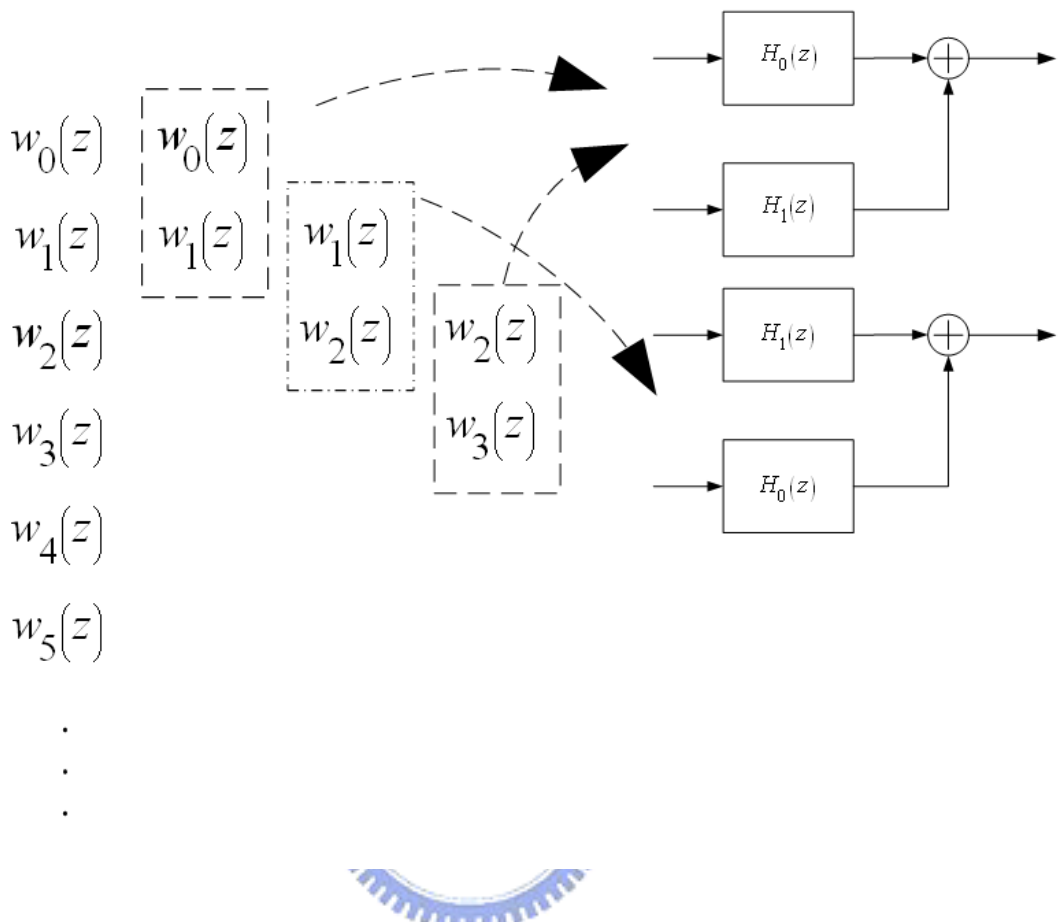


Figure 3-8 The rearrangement of polyphase component.

In Figure 3-8, the polyphase components are listed in the left of the figure, and the components are rearranged as the filter inputs. Here, the filters are also rearranged. Thus, this will lead to the structure in Figure 3-9.

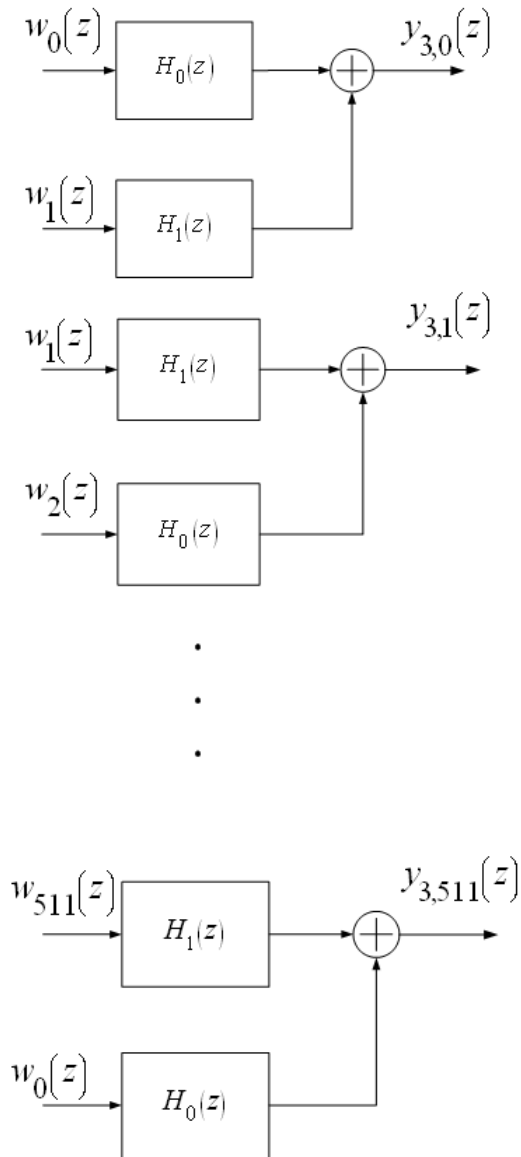


Figure 3-9 Rearranged polyphase structure in downsampling by 2.

In the Figure 3-9, we apply this idea to FS1016 where the stochastic codebook has 512 vectors. Different from previous polyphase structure in Figure 3-7, when $y_{3,k}(z)$ is synthesized, we can find that in this structure there is one branch which is the same with one of the branches when $y_{3,k-1}(z)$ was synthesized, except for

$y_{3,0}(z)$. For example, in Figure 3-7, we can see that

$$y_{3,0}(z) = w_0(z)H_0(z) + w_1(z)H_1(z) \quad (3.6)$$

and

$$y_{3,1}(z) = w_1(z)H_1(z) + w_2(z)H_0(z) \quad (3.7)$$

From both equations, it is clear that the $w_1(z)H_1(z)$ occurs two times in the synthesis of $y_{3,0}(z)$ and $y_{3,1}(z)$; however, we do not need to calculate $w_1(z)H_1(z)$ twice. When $y_{3,0}(z)$ is synthesized, we calculate the term $w_1(z)H_1(z)$ to generate $y_{3,0}(z)$; then, the term is stored for next iteration $k=1$ to generate $y_{3,1}(z)$. Therefore, in the duration of synthesizing $y_{3,1}(z)$, there is just one branch that need to calculate, thus this rearrangement further cut the computation down to half. Moreover, after this rearrangement, we do not gain the benefit of computation at the expense of quality.

However, after rearranging $w_r(z)$ and $H_f(z)$, the codebook vector $v_k(z)$ do not agree with its original polyphase components $w_{2k}(z)$ and $w_{2k+1}(z)$ in Equation (3.7). The reason is that the components $w_r(z)$ where $k = 0, 1, 2, \dots, \frac{N}{2} - 1$ are rearranged in the overlapped method. Thus, to get the benefits after rearrangement, the stochastic codebook must be modified to fit the structure in Figure 3-9. Therefore, the modified codebook vector $\hat{v}_k(z)$ is

$$\begin{cases} \hat{v}_k(z) = w_k(z^2) + z^{-1}w_{k+1}(z^2) & k = \text{even}, k \neq N-1 \\ \hat{v}_k(z) = w_{k+1}(z^2) + z^{-1}w_k(z^2) & k = \text{odd}, k \neq N-1 \\ \hat{v}_{N-1}(z) = w_{N-1}(z^2) + z^{-1}w_0(z^2) & k = N-1 \end{cases} \quad (3.8)$$

This is illustrated in Figure 3-10.

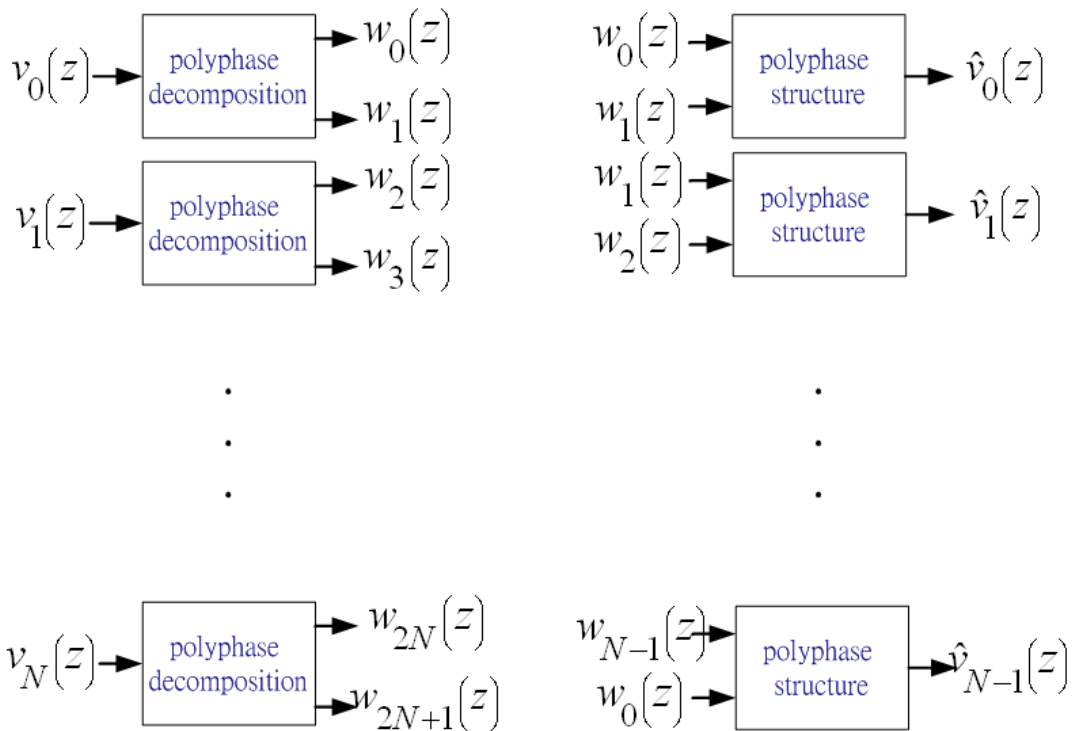


Figure 3-10 The comparison between original codebook and modified codebook.

Likewise, we can also apply this kind of structure in different downsample levels. For example, in downsampling by 3, we can re-arrange the polyphase components to form the modified codebook in Figure 3-11 and Figure 3-12.

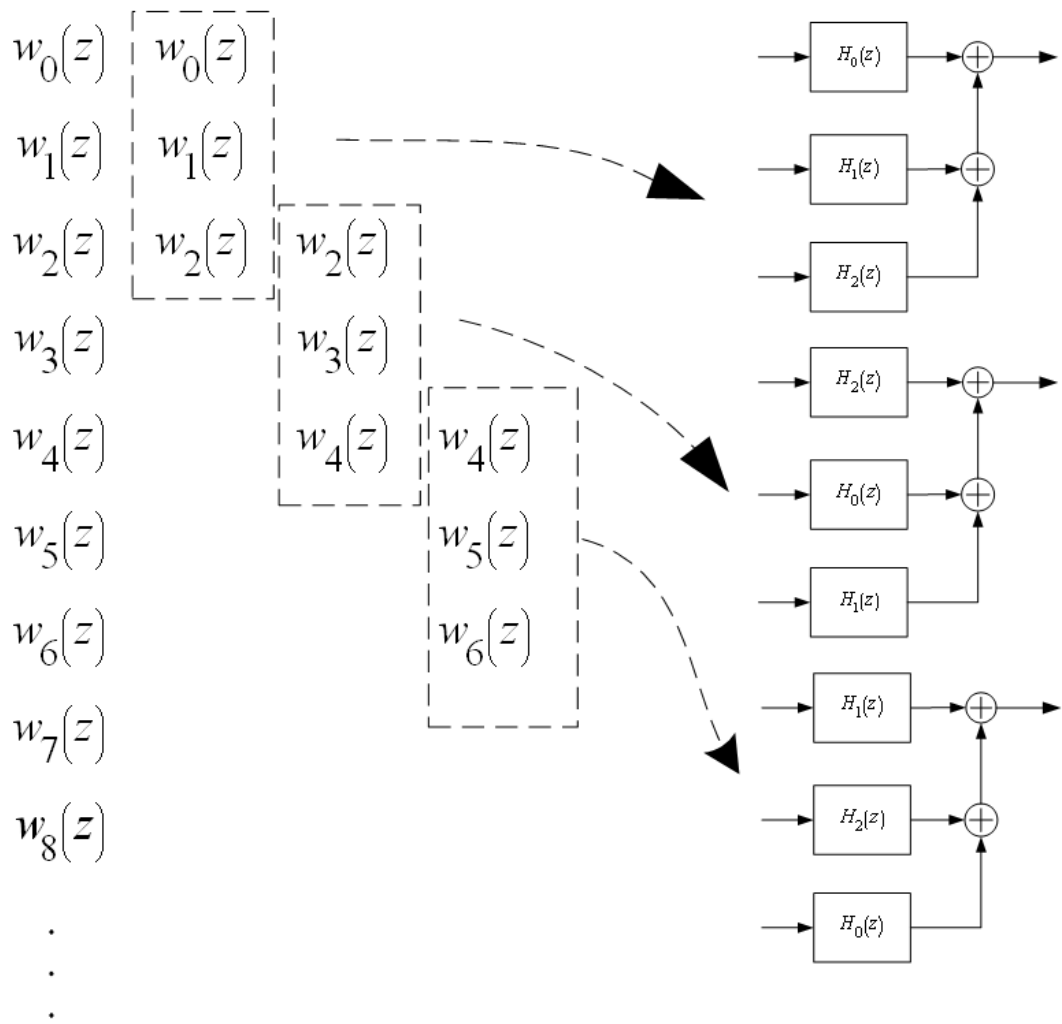


Figure 3-11 Rearranged polyphase structure in downsampling by three with one overlapping branch.

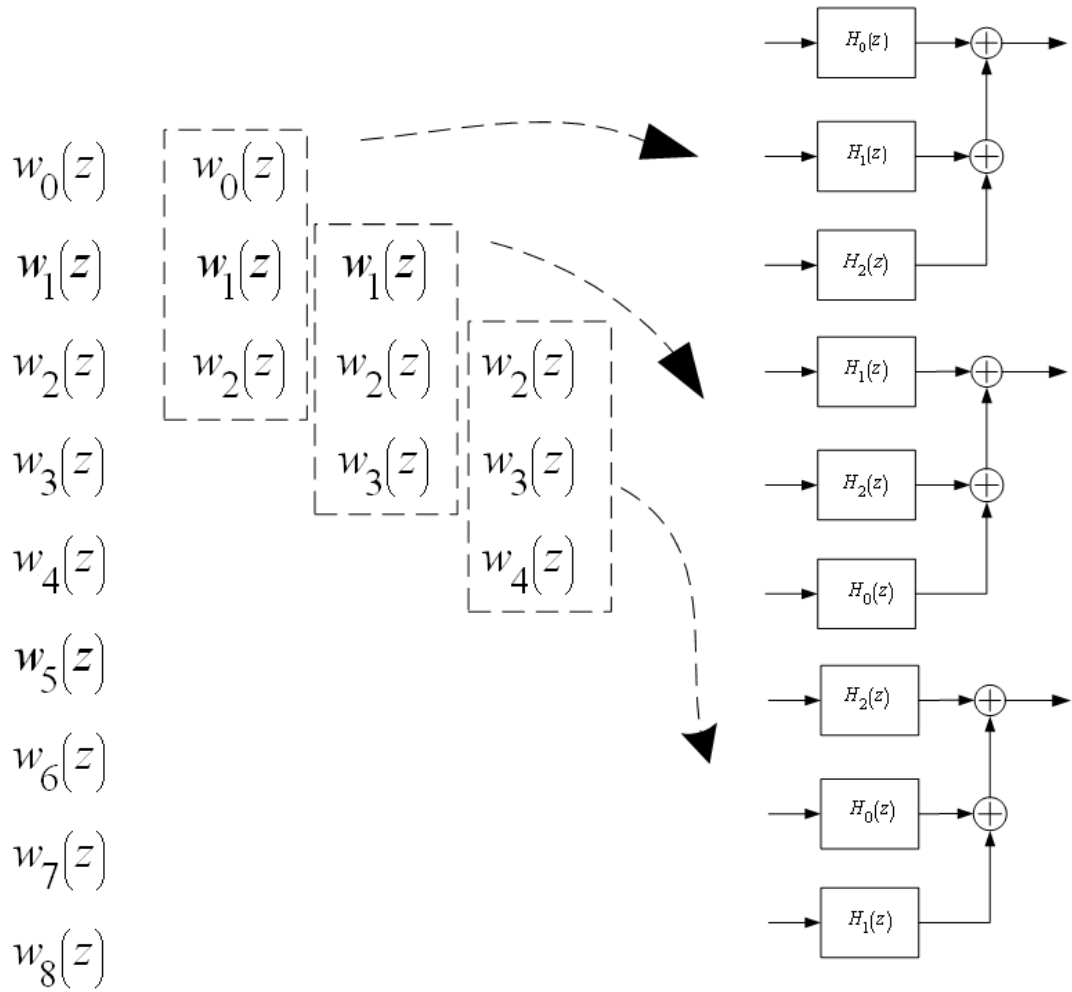


Figure 3-12 Rearranged polyphase structure in downsampling by three with two overlapping branch.

In the Figure 3-11, the modified structure with overlapping one branch is shown and the modified codebook vector $\hat{v}_k(z)$ is

$$\begin{cases} \hat{v}_k(z) = w_k(z^3) + z^{-1}w_{k+1}(z^3) + z^{-2}w_{k+2}(z^3) & k = 3m \\ \hat{v}_k(z) = z^{-2}w_k(z^2) + w_{k+1}(z^2) + z^{-1}w_{k+2}(z^3) & k = 3m + 1 \\ \hat{v}_k(z) = z^{-1}w_k(z^2) + z^{-2}w_{k+1}(z^2) + w_{k+2}(z^3) & k = 3m + 2 \end{cases} \quad (3.9)$$

Besides, the modified structure with overlapping two branches is shown in

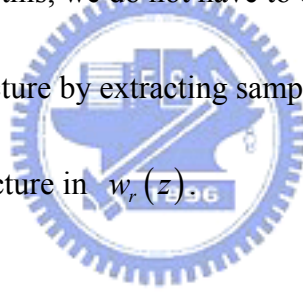
Figure 3-12 and the modified codebook vectors $\hat{v}_k(z)$ in donwsampling by 3 with

overlapping two vectors are

$$\begin{cases} \hat{v}_k(z) = w_k(z^3) + z^{-1}w_{k+1}(z^3) + z^{-2}w_{k+2}(z^3) & k = 3m \\ \hat{v}_k(z) = z^{-1}w_k(z^2) + z^{-2}w_{k+1}(z^2) + w_{k+2}(z^3) & k = 3m + 1 \\ \hat{v}_k(z) = z^{-2}w_k(z^2) + w_{k+1}(z^2) + z^{-1}w_{k+2}(z^3) & k = 3m + 2 \end{cases} \quad (3.10)$$

Of course, the structure with overlapping two branches will save twice computation than the one with overlapping one branch. Here, we do not let the last one and the first one to share the same computation.

However, in the implementations, we prefer to store $w_r(z)$ in the memory rather than $\hat{v}_k(z)$. Through this, we do not have to do the extra control operations to form the re-arranged structure by extracting samples from $\hat{v}_k(z)$ and can continue to use overlap-structure in $w_r(z)$.



3.3 Re-search method to increase hit rate

According to the previous definition, we classified the situation choosing the same codebook index as “hit” and choosing different codebook index as “miss”.

This is too arbitrary to assess the downsample method. As we have mentioned above, we can see the process of downsample as projecting the candidate vectors to

its subspace. The best result is that all the N dimension vectors projects to their subspace which can retain the vector that has maximum match both in N dimension and its subspace; as mentioned above, determining which the subspaces are optimal is a data-dependent process which induces more computation that will contrary to our goal of this thesis. However, although the downsample is not the optimal transformation to the vectors we want to manipulate, we can do the re-search method as a remedy to increase the probability of choosing the right vectors.

In the previous method, we choose the codebook index k that $m_k^{(l)}$ ranks number one among the $m_u^{(l)}$ where $u = 0, 1, 2, \dots, N-1$. As mentioned above, we refer the term “miss” to choosing the different codebook index k by using m_k and $m_k^{(l)}$; that is, we will obtain different results by choosing in dimension L and dimension l . However, this definition is rigid and not flexible. The reason is that most characteristics of vectors will retain after projecting it to its subspace, and the definition is so arbitrary that it classify the situation to miss only if the $m_k^{(l)}$ does not rank number one among $m_u^{(l)}$. To utilize the property of projection, we induce the re-search method that will fully take advantage of the retained characteristics so as to enhance the quality. Before introducing the re-search method, we have to

release the rigid definition of “miss”. In the previous definition, if the $m_k^{(l)}$ does not rank number one among the $m_u^{(l)}$ where $u = 0, 1, 2, \dots, N-1$ with k chosen in L dimension, we classify it to “miss”, even in the situation that $m_k^{(l)}$ rank number two. Thus, we redefine that when $m_k^{(l)}$ ranks outside from top t among $m_u^{(l)}$ is “miss” instead. This means that if the characteristics do not differ much after projection, we do not turn down it right away and classify it as “miss. To illustrate the reasons why we can take fully advantage of the retained characteristic of projection after redefinition, we do an experiment by downsample the vectors from 60 points to 30 points with $t = 10$, and the result is shown in Figure 3-13.

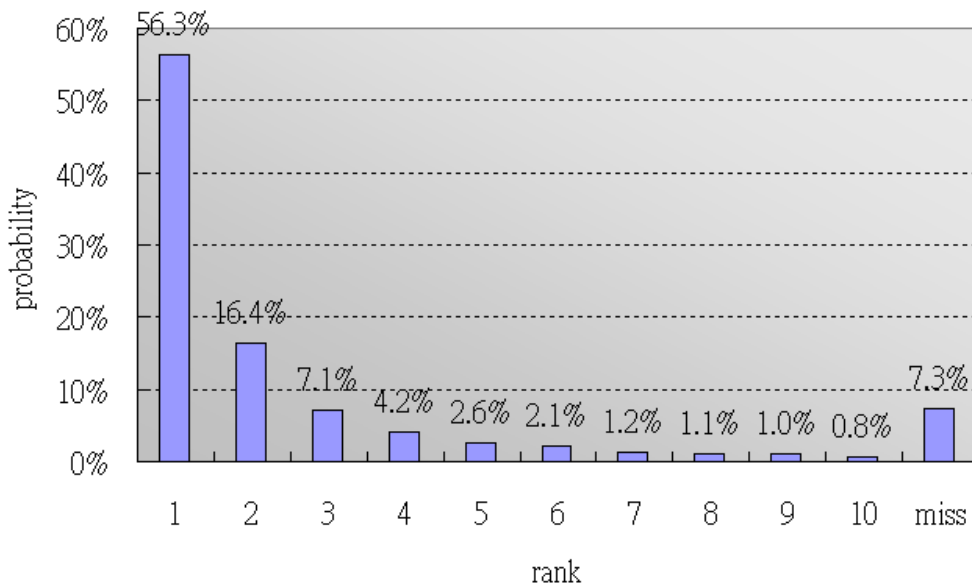


Figure 3-13 Probability of rank which $m_k^{(30)}$ fall in with the k chosen in dimension 60.

In Figure 3-13, we do the codebook search to obtain the codebook index k by m_k ; that is, we do the close-loop codebook search in L dimension first. By this codebook index k , the $m_k^{(30)}$ can be produce and then we can calculate that in what rank $m_k^{(30)}$ falls among the $m_u^{(30)}$ where $u = 0, 1, 2, \dots, N-1$ each subframe and the probability can be obtained. As shown in Figure 3-13, apart from the probability of “miss” (rank out of ten) and “hit” (rank number one), we have another 9 gray zones. Note that the total probability of “hit” and 9 gray zones is up to 92.7% which is far larger than the probability of “hit” 56.3%. That is why we say that the previous definition would not make use of the characteristics of projection fully, and why we induce the re-search method.

In short, the re-search method is that we will search the m_k again in dimension L after the searching in dimension l is done. In the beginning, we do the close-loop search in dimension l , so there are N codebook vectors that need to be synthesized and then we have to calculate the $m_k^{(l)}$, where $k = 0, 1, 2, \dots, N-1$. Next, the top t $m_k^{(l)}$ is picked up, so the corresponding t codebook vectors $\hat{v}_{\tilde{k}}$ are determined where $\tilde{k} = 0, 1, 2, \dots, t-1$. Finally, we do the search again just using the t codebook vectors $\hat{v}_{\tilde{k}}$ selected by last step in the L dimension, and the

codebook index can be determined by choosing the maximum $m_{\tilde{k}}$. With the codebook index k which is chosen in the L dimension, if $m_k^{(l)}$ rank top t among $m_u^{(l)}$, we can get the same codebook index k by re-search method. Taking Figure 3-13 for example, this probability is high (92.7%), so the probability of choosing the right codebook index is high. Thus, the quality is just a little worse than the quality generated by standard; that is, better than the quality produced just by the downsample method.

Generally speaking, we can see that this method enhance the probability of choosing the optimal codebook index k by inducing a little computation. In this method, we do the search N times in light-weighted computation to choose t candidate codebook index, and t times normal-weighted searching on these candidates are executed to choose the final codebook index resulting in inducing negligible quality loss compared to FS1016. Take the previous experiment in Figure 3-13 for examples, the probability getting the right codebook index is 92.7%, and there are just 512 light-weighted searching along with 10 normal-weighted search are need rather than 512 normal-weighted search in the standard. That is real bargain.

Moreover, because we do the re-search after the search in subspace (in fewer

dimension), so in the re-search procedure some computations are the same. So, the “normal-weighted computation” part will have less computation than standard. We will not elaborate here.

3.4 Classification before Stochastic Codebook Search

So far, we focus on discussing downsample method to reduce the computational load. The radical problem of downsample method is that we will choose the wrong codebook index after applying downsample method. Thus, we apply the re-search method as a remedy. However, the re-search is not always effective and may be a burden in computation. Like in the “miss” case that the index k is not in the candidate index \tilde{k} ; thus, re-search method is not only useless, but induces extra computation as well. Therefore, we want to know when re-search is effective or useless and in turn just apply the re-search method to the situation that this method is effective to get the right codebook index and eliminate the useless computation.

Here, we continue the experiment in Figure 3-13. In that experiment, we get “hit” and “miss” parts to do the experiments, but the 9 gray zone parts are excluded.

We analyze the spectrum of target signal $\mathbf{s}_{3,k}$ when the situation is “hit” and “miss”.

The results are shown in Figure 3-14.

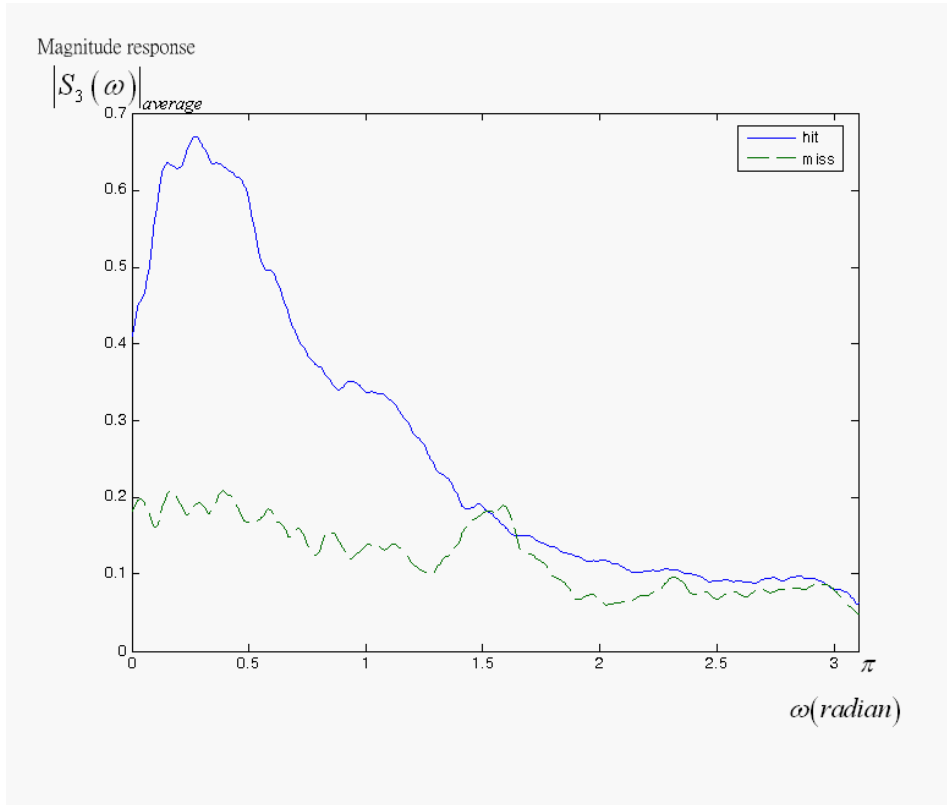


Figure 3-14 The spectrum of $\mathbf{s}_{3,k}$ when “hit” or “miss”.

From Figure 3-14, we can easily see that the magnitude spectrum of the target signal $|S_{av}(\omega)|$ in “hit” or “miss” differ obviously. The magnitude spectrum of the target signal in “hit” $|S_{av}(\omega)|_{hit}$ contains more energy in low frequency, while $|S_{av}(\omega)|_{miss}$ is white-noise-like.

That is an important result, because we can know when the situation tends to be “hit” or tends to be “miss” by observing the spectrum of incoming target signal.

Therefore, we can use the re-search method when the situation tends to be “hit”, and

close it when the situation tends to be “miss”. Thus, by applying the classification we can avoid inducing extra useless computation.

However, why will lowpass-like signal tend to be “hit” after projection, while while-noise like signals tend to be “miss”? In the later chapter, we will do the same experiments on the dimension 30 and 20 (downsample two and downsample three), and the result is like Figure 3-13; that is, we get a lowpass signal in the “hit” situation. Besides, the results show the tendency that the more we downsample, the lower spectrum the energy is allocated in.

From these experiments, we guess that the reasons why we can use downsample method in replace of the standard search are that we just compare the lower part of the spectrum which contains most of the characteristics of the speech. From basic DSP theory, the signals will expand the spectrum to M times after downsample M . Thus, if the aliasing in higher frequency after downsample is negligible, we can compare the signals after the spectrum expansion which means comparing the lowpass part before downsample and the reduction in complexity can achieve.

However, this conjecture may be hazardous, because there is aliasing in spectrum after downsample, so we compare the signals with that; moreover, the lowpass-like signals does not always lead to “hit”. Strictly speaking, they just result in higher

probability to be “hit”. So, the reasons behind this result are need to further research.

So far in this Section, we know that we can execute the classification to decide whether we apply the re-search method or not. Now, we have to investigate the issue that how to implement the classification.

In some applications, some systems will apply voiced/unvoiced classification. In most situations, the voiced signals will lead to the low-pass signal. Thus, we can utilize these kinds of functions to do the classification.

However, if there is no voiced/unvoiced classification function in the CELP system, we can do the classification by comparing how much the target signal $s_{3,k}$ differs between the energy in lower frequency and higher frequency. If the energy in lower frequency is much larger than the higher frequency beyond a specific threshold, we classify it as lowpass-like signal and we apply the downsample along with re-search method to close-loop search in this subframe. While, if the energy does not differ much (below the specific threshold), we view it as noise-like signal, and we do the close-loop search as it does in standard. This can be shown in Figure 3-15.

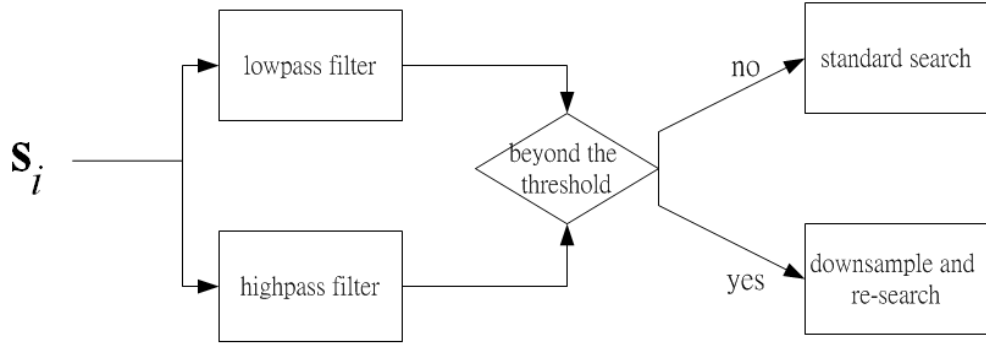


Figure 3-15 Flow chart of classification.

To get the lower frequency part and higher frequency part of $s_{3,k}$, we add one lowpass filter $g_{low}[n]$ and one highpass filter $g_{high}[n]$ to our system and get two new signal:

$$s_{low,k}[n] = s_{3,k}[n] * g_{low}[n] \quad (3.11)$$

$$s_{high,k}[n] = s_{3,k}[n] * g_{high}[n] \quad (3.12)$$

By these two signals, we can classify the situation as “hit” if the energy difference is much larger than a threshold c . We can implement it as

$$\sum_{n=0}^{L-1} |s_{low,k}[n]| >_{hit} c \sum_{n=0}^{L-1} |s_{high,k}[n]| \quad (3.13)$$

Here, we adopt absolute value instead of square for low complexity.

Assume that the lowpass filter $g_{low}[n]$ and $g_{high}[n]$ are of order G . So, we add $2GL$ multipliers and $2L$ additions per loop to achieve classification. The

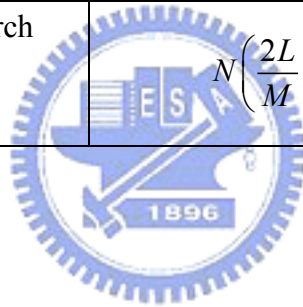
filter order G and the computation load is much smaller than close-loop search

when we choose the reasonable the filter order G .

The computation load of the method we mentioned is summarized in Table 3-2.

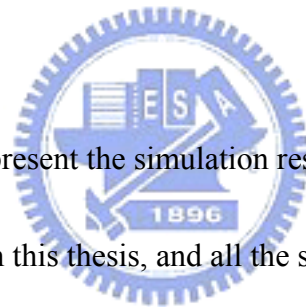
Table 3-2 Computation load in classification, standard search and downsample along with re-search.

Function	Multipliers
Classification	$2GL$
Standard search	$N(2L+1)$
Downsample before search and re-search	$N\left(\frac{2L}{M}+1\right)+t(2L+1)$



Chapter 4

Simulation Results



In this chapter, we will present the simulation results and listening test in the methods that we proposed in this thesis, and all the simulations are done based on the FS1016 standard with some required modifications.

In the Section 4.1, downsample two, three and four are applied. In the Section 4.2, the comparison of different method in signal-noise-ratio (SNR), listening test and computational load is presented.

4.1 Comparison in Downsample two, three and four

In the Chapter 3, the hit rate of different points is shown in Figure 3-2. Here, we just present downsampling in two, three and four. As mentioned above, the more we downsample in target vectors and candidate vectors, the more computation we can save; however, the more degradation in speech quality we sacrifice too.

Here, we focus on the distribution in rank, and the spectrum of target signal $s_{3,k}$ when the situation is hit.



4.1.1 Downsample Two

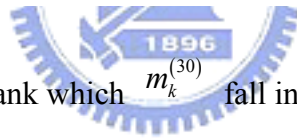
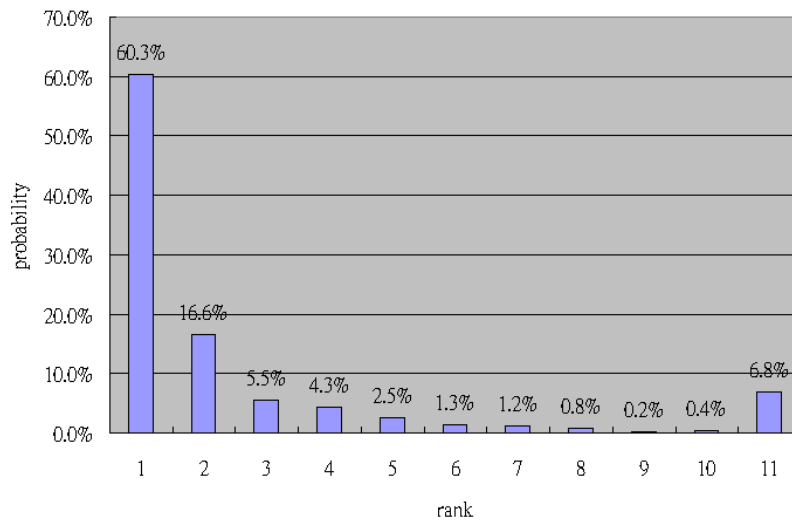


Figure 4-1 Probability of rank which $m_k^{(30)}$ fall in where the k is chosen in dimension 60.

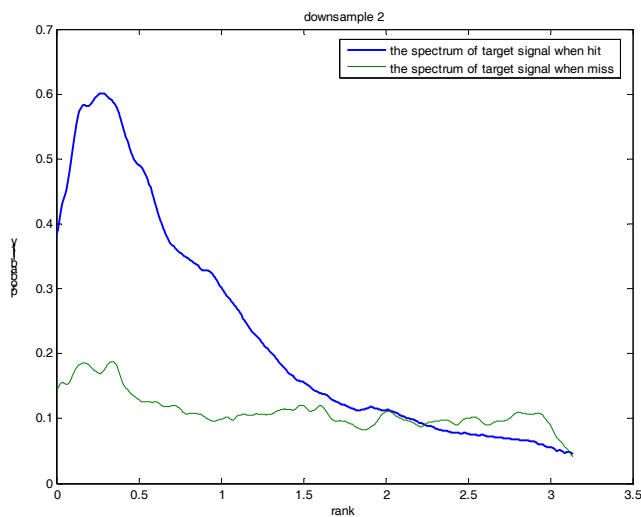


Figure 4-2 The spectrum of $s_{3,k}$ when “hit” or “miss” in downsampling 2.

4.1.2 Downsample Three

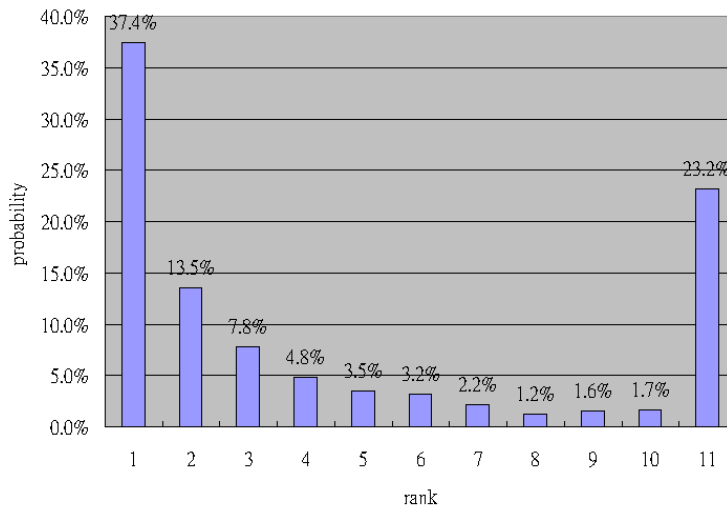


Figure 4-3 Probability of rank which $m_k^{(20)}$ fall in where the k is chosen in dimension 60.

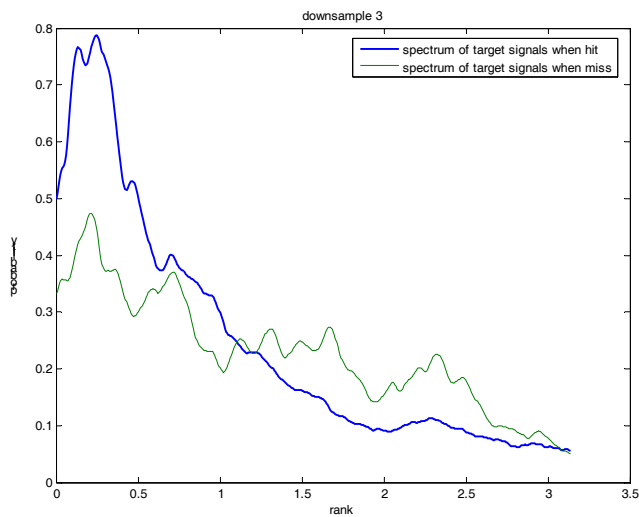


Figure 4-4 The spectrum of $s_{3,k}$ when “hit” or “miss” in downsampling 3.

4.1.3 Downsample Four

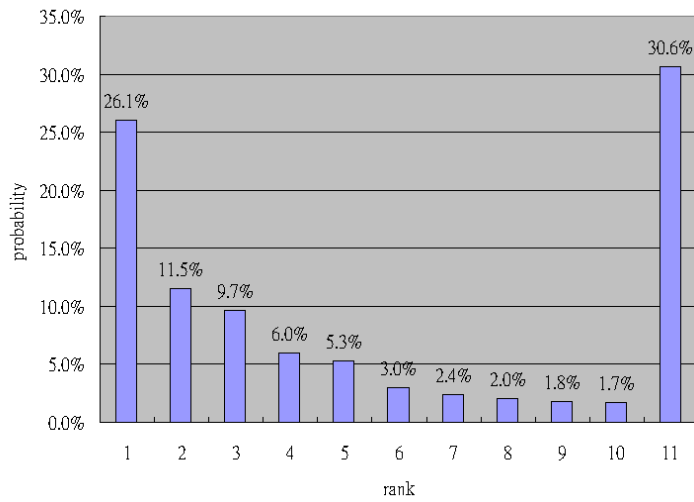


Figure 4-5 Probability of rank which $m_k^{(15)}$ fall in where the k is chosen in dimension 60.

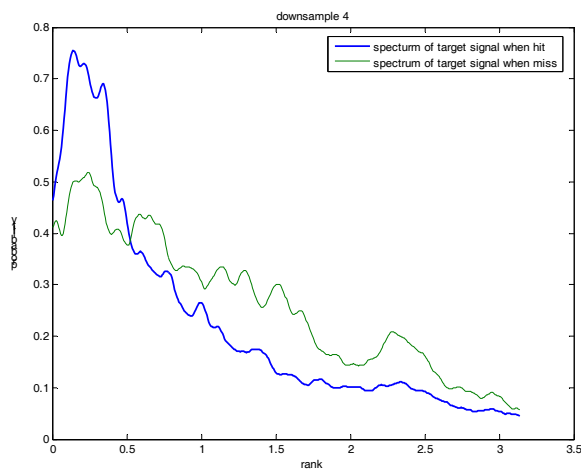


Figure 4-6 The spectrum of $\mathbf{s}_{3,k}$ when “hit” or “miss” in downsampling 4.

4.1.4 Discussion

From above results, we can see that the hit rate decreases with the increase of the downsample level. Besides, in the cases of downsample three and downsample four, the differences between the characteristics of the spectrums of target signals when hit and miss are not apparent as that in the downsample two. This means that we have to use sharper filters during classification.



4.2 Comparison between different proposed method

In this section, different methods are compared. We design four situations (situation A, B, C, D) to do the comparisons.

In the situation A, all the computations are the same with standard FS1016. In the situation B, the close-loop search in stochastic codebook part is downsampled two. In the situation C, the modified codebook with downsample two is used, and in the situation D, the classification with re-search method is added in situation B. This summarized in Table 4-1.

Table 4-1 Applied method in different situation.

Method Situation	Downsample by 2	Modified codebook	Classification	Re-search
A				
B	○			
C	○	○		
D	○		○	○

And speeches samples in three males (m1, m2, m3) and females (f1, f2, f3) are used. The results are shown in Figure 4-7 and Table 4-2.

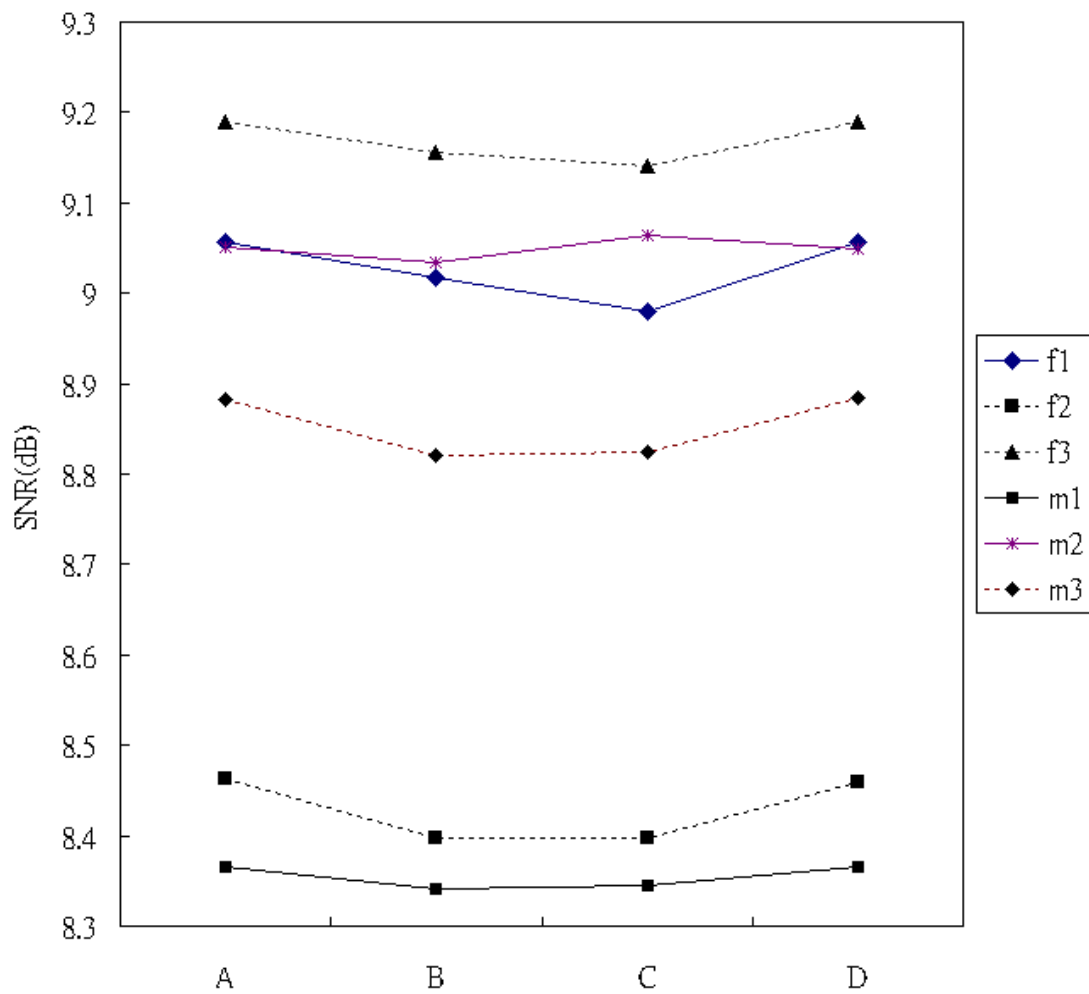


Figure 4-7 The SNR in different methods.

Table 4-2 The SNR(dB) in different methods.

	f1	f2	f3	m1	m2	m3
A	9.05	8.46	9.18	8.36	9.05	8.88
B	9.01	8.39	9.15	8.34	9.03	8.82
C	8.97	8.39	9.14	8.34	9.06	8.84
D	9.05	8.45	9.18	8.36	9.04	8.88

As shown in Figure 4-7, the situation B suffers a little degradation, while situation D can improve the speech quality that is degraded by downsample.

Here, we use comparison category rating (CCR) to evaluate the subjective speech quality of these methods [3]. Twenty listeners write down their scores in comparison with the situation A (FS1016 only) using the scale shown in Table 4-1.

And the average score is shown in Table 4-3.

Table 4-3 CCR scale.

Comparison	Score
Much better	+3
Better	+2
Slightly better	+1
About the same	0
Slightly worse	-1
Worse	-2
Much worse	-3

Table 4-4 The averaged scores of CCR in comparison of situation A (standard).

Situation Speech samples	B	C	D
m1	0	-0.1	0
m2	0	-0.1	0
m3	-0.1	0	0
f1	0	0	0
f2	0	-0.1	0
f3	0	0	0

We can see that the subjective speech quality decreases a little due to downsample by 2. Moreover, the quality will be worse by applying modified stochastic codebook. However, listeners often can not tell the difference between situation A and situation D.

Besides, the computational load is also shown in Figure 4-8.

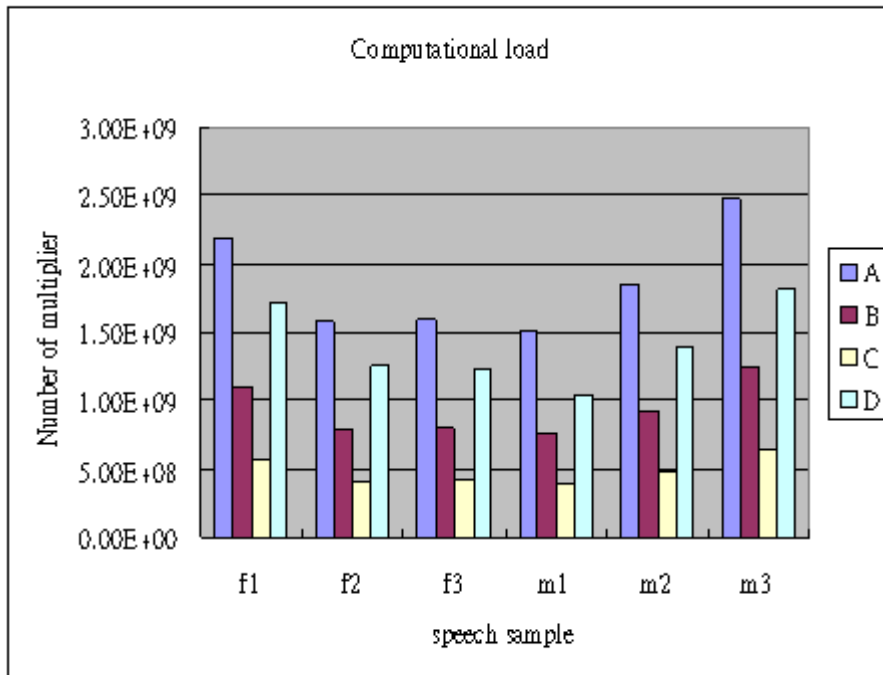
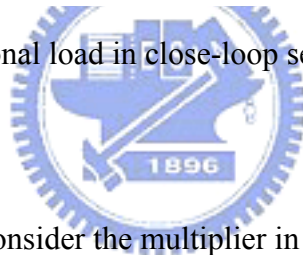


Figure 4-8 The computational load in close-loop search.



In Figure 4-8, we only consider the multiplier in the close-loop search. From this figure, we can easily see that the computational load between different methods we propose.

Here, we show more detailed information between methods. In the Figure 4-9, the hit rate about situation B and D is shown.

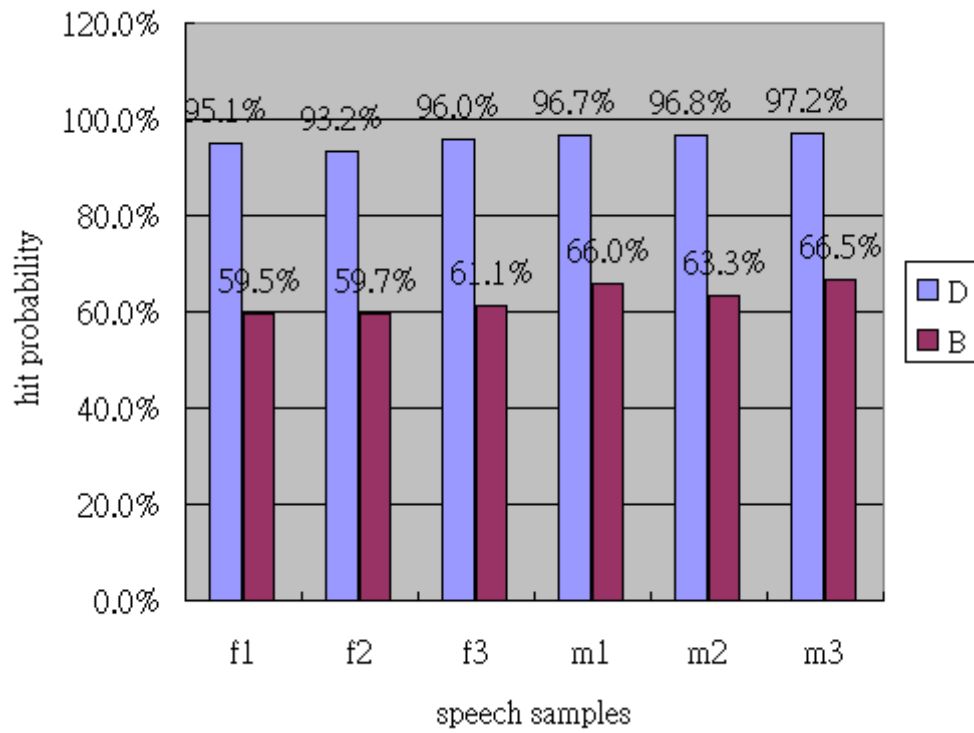
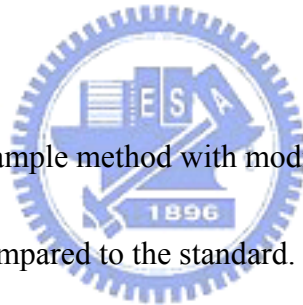


Figure 4-9 The probability of hit in situation B and D.



Chapter 5

Conclusion and Future work



In this thesis, the downsample method with modified codebook we proposed can save a lot of computation compared to the standard. Besides, the quality issue in these methods has been considered and solved. In re-search and classifying scheme that we propose as a remedy to downsample method can increase the quality that nearly the same with the quality of the standard FS1016, but the computational load that needs to add is low. To sum up, we can fairly say that by applying these methods the total computational load can be reduced at the expense of imperceptible degradations.

However, the philosophy behind the downsample method is not very clear yet. Thus, the researches are limited to the statistic analysis. Besides, the verifications in

applying these techniques to other codebook structures are not done yet. Thus, there is still much space that is worth being investigated



Bibliography

[1] WAI C. CHU “Speech coding algorithms,” in 2003 by John Wiley & Sons.

[2] Masahiro Serizawa, Hironori Ito and Toshiyuki Nomura, “A silence compression algorithm for multi-rate/dual-bandwidth MPEG-4 CELP standard,” in IEEE 2000.

[3] Lucio Martins da Silva and Abraham Alcaim “A modified CELP model with computationally efficient adaptive codebook search,” IEEE 1995.

