The Implementation and Optimization of AMR on Mobile Device

Jie Yang
Department of Computer Science, Huazhong University of Sci&Tech, Wuhan, China
Email: youngth98@gmail.com

Sheng sheng Yu and Jingli Zhou
Department of Computer Science, Huazhong University of Sci&Tech, Wuhan, China
Email: { ssyu, jlzhou }@mail.hust.edu.cn

***Abstract*—Considering the limited resource of mobile device, the operation complexity of the applications needs to be reduced. This paper presents the implementation of Adaptive Multi Rate (AMR) on mobile device and especially focuses on reducing the computational complexity. AMR is an algebraic code excited linear prediction (ACELP) based algorithm which could achieve satisfying speech quality by using the analysis-by-synthesis method. It has been chosen by the Third Generation Partnership Project (3GPP) as the mandatory codec for the third generation (3G) cellular systems. The optimizations include algorithm and instruction level. Based on the mobile device i-mate JAMin which has the ARM processor operating at 195 MHz, one second speech costs only 0.6 second processing time. So, more resource is reserved for other applications.**

***Index Terms*—AMR, Speech Coder, Optimization, Mobile Device**

I. Introduction

There are more and more applications developed on mobile device, such as audio and video processing applications. The requirement may be high, such as space, processing time, energy and so on. But the resource of mobile device is very limited. So, real-time audio and video communication needs efficient optimization about the compression algorithm on power-efficient platform.

Some researchers focus on the implementation and optimization of video encoder on mobile device. Optimization strategies are proposed based on the context of the scenes and reduced more than 40% of computational complexity of H.264 [1]. Different optimization techniques in motion estimation, intra prediction process and other major modules of H.264 encoder were discussed in [2]. Real time encoding with QCIF (176×144) resolution on the mobile device was realized on a kind of Personal Digital Assistant (PDA) which has a 624MHZ frequency processor [3].

This paper presents the implementation and the optimization of Adaptive Multi-Rate (AMR) on mobile phone which has the ARM processor, including algorithm and instruction modification. The experiment have demonstrated that the methods reduced the complexity of computation greatly with little sacrificing of speech quality.

The rest of this paper is organized as follows: In Section Ⅱ, the AMR standard and the system is briefly reviewed. Sections Ⅲ presents the optimization methods in details. Experiment and evaluation results are shown in Section Ⅳ and the conclusion is drawn in Section Ⅴ.

II. Overvie of AMR and The System

The AMR speech is based on Code Excited linear prediction (CELP). It supports 8 encoding modes with bit rates between 4.75 and 12.2 kb/s [4].The AMR speech coder operates for each speech frame of 20ms (160 samples) at the 8 kHz sampling frequency. Two pre processing functions are applied prior to the encoding process: high pass filtering and signal down scaling. After linear prediction (LP) analysis, the LP parameters are converted to line spectrum pairs (LSP) and jointly quantized. The open-loop pitch search is performed once per frame for the 4.75 and 5.15 kbps modes. While for all other modes, it is performed twice per frame. The frame is then divided into 4 subframes. An open loop pitch lag is estimated in every other subframe based on the perceptually weighted speech signal and used for the adaptive codebook search. Closed loop pitch analysis is performed by searching around the open loop pitch lag. The gains of the adaptive and fixed codebook are quantified. Finally, the filter memories are updated for finding the target signal in the next subframe [4].The encoding scheme of each speech mode is almost the same, while bit frame is a little different. In each 20ms speech frame, 95, 103, 118, 134, 148, 159, 204 or 244 bits are produced, corresponding to each bit-rate from 4.75 to 12.2 kb/s [5]. The encoding block diagram is shown in Fig. 1.

Right now, many mobile phones and personal digital assistants adopt ARM processor which is designed specifically for limited memory and low power consumptionl [6]. The ARM architecture is generally a 32-bit RISC processor architecture developed by ARM Limited that is widely used in embedded designs. ARM uses the techniques like variable execution time, subword parallelism, digital signal processor-like operations, thread-level parallelism, exception handling and multiprocessing to enhance its performance [7].

© 2009 ACADEMY PUBLISHER
doi:10.4304/jsw.4.9.984-991

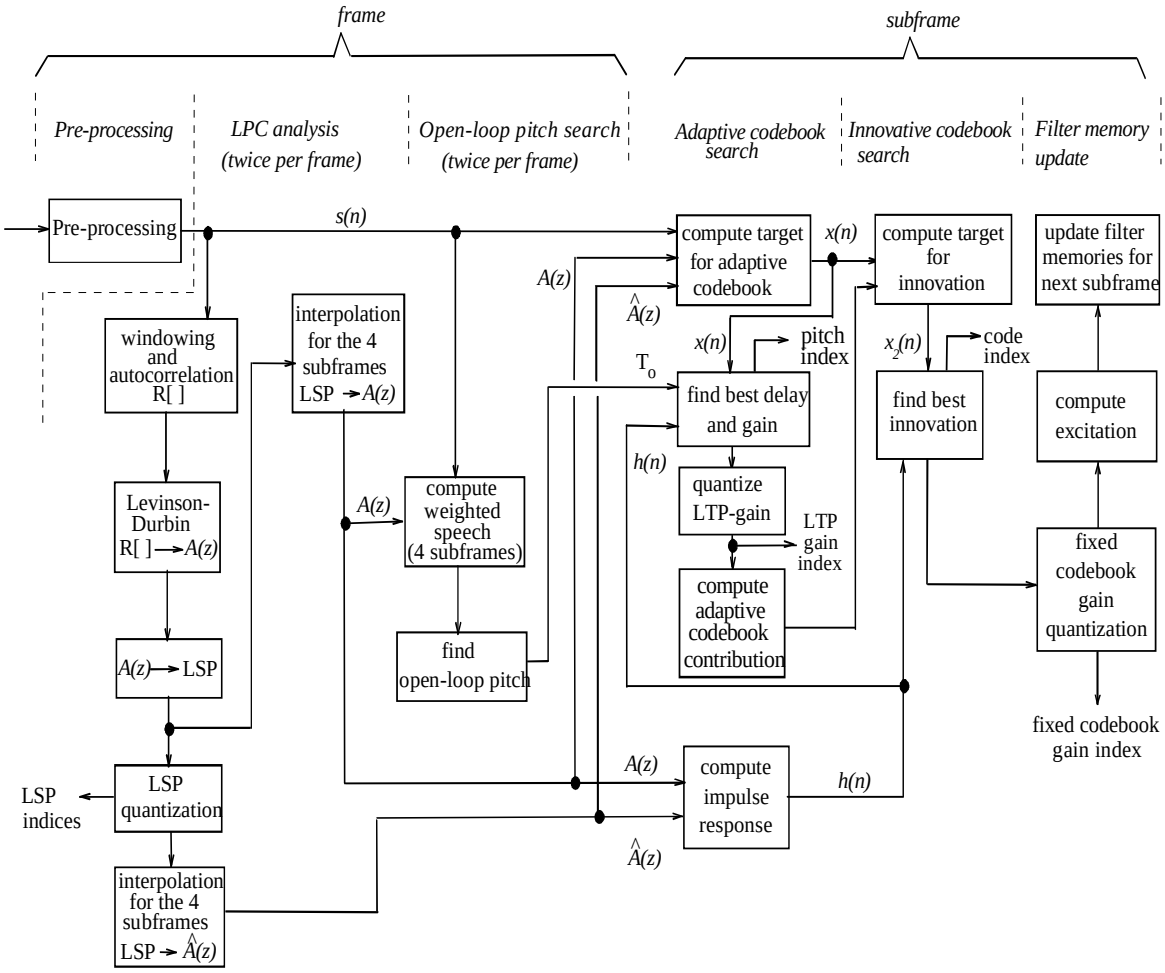

Fig. 1. Simplified block diagram of the adaptive multi-rate encoder

The ARM instruction set differs from the pure RISC definition in several ways that make it suitable for embedded applications [8], such as:

- Variable cycle execution for certain instructions.
- Inline barrel shifter leading to more complex instructions.
- Thumb 16-bit instruction set.
- Conditional execution.
- Enhanced instructions.

The mobile phone we use is i-mate JAMin. The main processor of the phone is OMAP 850 produced by Texas Instruments (TI) which combines an ARM926 and ARM7 general-purpose RISC processor. The processor operates at 195 MHz, for high performance with low power consumption.

We focus on development environment (IDE) packages: Microsoft's Visual Studio .NET2005 for developing WinCE applications. To synchronize the phone with personal computer (PC), the software ActiveSync is installed. The information could be exchanged between the mobile phone and PC in this way.

III. IMLEMENTATION AND OPTIMIZATION

The codec from the 3GPP has been used as the foundation model of the encoder. We transplanted the codec with Microsoft's Visual Studio.NET 2005. The screenshot of the mobile phone is shown in Fig. 2.

Fig. 2. Hospital system screen

© 2009 ACADEMY PUBLISHER

Since the computation complexity of the original codec is high, it needs to be optimized to get high performance. Firstly, we measure the execution time of every module. Different modes of AMR have different degrees of complexity, in which 12.2 kb/s mode has the maximum complexity. The execution time of each sub-block is listed in Table Ⅰ. From Table Ⅰ, we can see that the fix codebook search is the most time-consuming part. Linear predictive coding, close-loop pitch analysis and open-loop pitch analyses also take a certain amount of time. The optimization includes the following aspects:

A. Algorithmic Optimization

1) Open loop pitch analysis: Open-loop pitch analysis is performed in order to simplify the pitch analysis and confine the closed-loop pitch search to a small number of lags around the open-loop estimated lags [5]. During the search process, the maxima of the correlations $R(k)$ are calculated:

$$R(k)=\sum_{n=0}^{79} s_w(n)s_w(n-k) \qquad (1)$$

While the range of k has been divided into three parts: 1) k=72,...,143 , 2) k=36,...,71 , 3) k=18,...,35. The three maxima $R(t_i)$ where t_i is the lag value corresponding to the maxima of each part are normalized by dividing

$$\sqrt{\sum_n s_w^2(n-t_i)} \qquad (2)$$

The value t_i that maximizes the equation (3) will be regarded as the pitch lag.

$$R'(t_i)=\frac{O(t_i)}{\sqrt{\sum_n s_w^2(n-t_i)}} \qquad (3)$$

Among the three pith lags in different range, the lower clause one is selected as the final values in order to avoid choosing pitch multiples. From the above analysis, we can see that the searches one by one of the whole scope would take considerable amount of computation.

To reduce the complexity, we can modify the search process into two steps. In the first step, the weighted speech signal $s_w(n)$ is re-sampled in accordance with the ratio of 4:1 to get $s_{w-new}(n)$ and the search scope of t_i changes from 5 to 35. The $s_{w-new}(n)$ is searched one by one to find the value $t_{\max}$ that maximize equation (3). In the second step, $s_w(n)$ is searched among the new range $t_i \in [4*t_{\max}-3, 4*t_{\max}+3]$, and the value t_i that maximize equation (3) will be regarded as the final pitch lag. In this way, the scope of the search is reduced and the times of searching are greatly decreased.

2) Fix codebook search: The traditional fix codebook search is to get the impulse code which maximizes the following equation (4):

$$A_k=\frac{(C_k)^2}{E_{D_k}}=\frac{(d^t c_k)^2}{c_k^t \Phi c_k} \qquad (4)$$

where $d=H^t x_2(n)$ is the correlation between the target signal $x_2(n)$ and the impulse response $h(n)$. H is a lower triangular Toepliz convolution matrix with diagonal $h(0)$ and lower diagonals $h(1),...,h(39)$, and $\Phi=H^tH$ is the matrix of correlations of $h(n)$. The vector d (backward filtered target) and the matrix Φ are computed prior to the codebook search [5].

We still take the 12.2 kb/s mode as the example. For this mode, 40 positions in a subframe are divided into 5 tracks, where each track contains two pulses, as shown in Table Ⅱ. AMR adopts the depth first tree search method. The first pulse $i0$ is always set at the position corresponding to the global maximum value and the pulse $i1$ is set as the local maximum of one track. The search is performed in five nested loops, where in each loop the contribution of a new pulse is added. For example, T2–T3 is searched for the next two pulses, followed by T4–T5, T6–T7, T8–T9, T10–T11 searches consequently. The final 8-pulse code vectors are determined with (8 × 8) × 4 searches. Later, the starting positions of the 9 pulses are cyclically shifted. During the process, the local maximum of a different track is always set as $i1$. So, the whole search times are (8 × 8) × 4× 4=1024. The disadvantage of two-track-based sequential search is that once a certain pulse is selected, the second pulse will be restricted in the search loop and results in redundant computation complexity.

TABLE Ⅰ
THE DETAILED EXECUTION TIME OF EACH SUB-BLOCK (12.2 kb/s)

Function block	Processing time(ms)
Pre_process	337
Linear predictive coding	1122
Line spectral pair processing	5622
Open-loop pitch analysis	4981
SubframePreProc	4120
Close-loop pitch analysis	5415
Fix codebook search	12178
Gain_quant	1211
Memory update, etc.	2307
Total	37293

TABLE Ⅱ
POSITIONS OF PULSES FOR THE TRACKS IN THE ALGEBRAIC CODEBOOK (12.2 kb/s)

Track	Pulse	Positions
1	i0, i5	0, 5, 10, 15, 20, 25, 30, 35
2	i1, i6	1, 6, 11, 16, 21, 26, 31, 36
3	i2, i7	2, 7, 12, 17, 22, 27, 32, 37
4	i3, i8	3, 8, 13, 18, 23, 28, 33, 38
5	i4, i9	4, 9, 14, 19, 24, 29, 34, 39

© 2009 ACADEMY PUBLISHER

The implementation of the AMR speech coder on the 16 bit fixed-point DSP was described in [9]. A modified depth first tree search method used in algebraic codebook search was also referred to. We adopt that method and each pulse is searched in the track separately to avoid the nested layers of the cycle. The times of searching by the new method are (8+8)*4*4=256 and reduced 75% computation complexity.

For other modes of AMR, the search method could be modified with the same approach and the computation load reduces from $O(n^2)$ to $O(n)$.

B. Code Optimization

1) ARM instruction optimization: We also optimize the system by using ARM instruction since the mobile phone has the ARM processor core. The enhanced instructions of ARM could perform some functions more efficiently.

For example, L_add is the function that used for the addition of the two 32 bits variables with overflow control. In the original system, it is written in C language. The code contains a number of judgment processes and is relatively time-consuming [10].

We use

$$QADD \quad r0, \ r0, \ r1 \qquad (5)$$

to realize the function and this avoids the requirement for any additional code to check for possible overflows [6].

For other functions, such as add, sub, div_s, L_add, L_mac,L_msu, L_mult, L_sub, norm_l, norm_s, saturate, we have also replaced them by using ARM-based instructions to save running time. Some are listed below:

L_mac:

$$\begin{aligned} &SMULBB \quad r2, \ r1, \ r2 \\ &QDADD \quad r0, \ r0, \ r2 \\ &BX \quad LR \\ &END \end{aligned} \qquad (6)$$

L_msu:

$$\begin{aligned} &SMULBB \quad r2, \ r1, \ r2 \\ &QDSUB \quad r0, \ r0, \ r2 \\ &BX \quad LR \\ &END \end{aligned} \qquad (7)$$

L_mult:

$$\begin{aligned} &SMULBB \quad r1, \ r0, \ r1 \\ &MOV \quad r0, \ r1, \ lsl \#1 \\ &BX \quad LR \\ &END \end{aligned} \qquad (8)$$

L_sub:

$$\begin{aligned} &CMP \quad r1, \ \#0 \\ &MVNEQ \quad r1, \ \#0\text{x}80000000 \\ &SUBEQ \quad r0, r1, r0, \ asr \ \#31 \\ &SWINE \quad 0\text{x}06000 \\ &BX \quad LR \\ &END \end{aligned} \qquad (9)$$

Saturate:

$$\begin{aligned} &LDR \quad r1, \ =0\text{x}00007FFF \\ &MOV \quad r2, \ r0, \ ASR \#15 \\ &TEQ \quad r2, \ r0, \ ASR \#31 \\ &EORNE \quad r0, \ r1, \ r0 \ ASR \#31 \\ &BX \quad LR \\ &END \end{aligned} \qquad (10)$$

2) Remove Redundant Operations: We find that some operation in the loop could be moved outside. The amount of calculation can be reduced while the result would not be affected. For example, assuming L_CODE is a constant and Eq. (11) could be replaced by Eq. (12). The subtraction operation in the loop can be avoided. :

```
for( ; ; )
{...
   j = L_CODE −1;
      for( ; ; j−− )
         {...
            }
  ...
}                                   (11)

s = L_CODE −1;
for( ; ; )
{...
   j = s;
      for( ; ; j−− )
         {...
            }
  ...
}                                   (12)
```

© 2009 ACADEMY PUBLISHER

There are many functions containing loop part. In some cases, the counter increases to some arbitrary limit when the loop stops. It is suggested that the loop counter count down to zero [7]. This is because the comparison with zero is free since the result is stored in the condition flags.

3) Reduce the complexity of operations: The left or right shift operation can be used instead of multiplication or division. Usually, divided or multiplied by two could be completed by the shifted to right or left.

In fact, multiplied by any integer can be used to replace by addition shift and multiplication. The implementation time of shift and addition is less than multiplication instructions. For example, Eq. (13) could be replaced by Eq. (14).

$$i = i \times 9 \tag{13}$$

Could be replaced by

$$i = (i << 3) + i \tag{14}$$

4) Loop unrolling: For ARM7 or ARM9 processors, the subtract takes one cycle and the branch three cycles, giving an overhead of four cycles per loop [7]. We can unroll some loops to reduce the computation cycles. Parallelism can be exploited by using pipelined access to such data structures. We need to mention that if cycles of the loop take a very small proportion, the effect of loop unrolling will not be effective.

C: Complier Optimization

We have exploited Visual Studio 2005 compiler. It provides 32-bit C/C++ compiler used to compile and link 32-bit ARM C, and C++ programs [11]. The complier option includes:

- /Od: turns off all optimizations in the program and speeds compilation.
- /O1: creates the smallest code in the majority of cases.
- /O2: creates the fastest code in the majority of cases. (default setting for release builds)
- /Ox: combines optimizing options to produce code that favors execution speed over smaller code size.
- Ob0: Disables inline expansion, which is on by default.
- /Ob1: expands only functions marked as inline, __inline, __forceinline or__inline or, in a C++ member function, defined within a class declaration)/ob2(Expands functions marked as inline or __inline and any other function that the compiler chooses.
- Ob2: Expands functions marked as inline or __inline and any other function that the compiler chooses (expansion occurs at the compiler's discretion, often referred to as auto-inlining).
- /Oi: replaces some function calls with intrinsic or otherwise special forms of the function that help your application run faster. Programs that use intrinsic functions are faster because they do not have the overhead of function calls, but may be larger because of the additional code created.
- /Ot: maximizes the speed of EXEs and DLLs by instructing the compiler to favor speed over size. (This is the default.) The compiler can reduce many C and C++ constructs to functionally similar sequences of machine code.
- /GL: enables whole program optimization.

Since we hope to achieve faster speed, so /O2, /Ob2, /Ot, /GL are selected from relative options.

Ⅳ. System Test and Analyze

The testing speech materials we use are 30 seconds long, 8000 sample rate, mono channel and 16-bit word signals.

The processing time with different optimization strategy for all modes is presented in TableⅢ and the comparison of time is shown in Fig. 3. If we use the complier optimization only, the 30 seconds speech cost about 35 seconds processing time on the mobile device. While with all the methods, the time have reduced to nearly 18 seconds and save nearly half time.

TABLE Ⅲ
THE COMPARISON OF PROCESSING OF ALL MODES

Mode	Time(ms)		
	With compiler optimization only	Adding algorithmic optimization	With all optimizations
MR122	37293	24613	18675
MR102	35651	23748	18222
MR795	35986	23917	18240
MR74	33502	22244	17380
MR67	35349	22267	17384
MR59	30700	20705	16053
MR515	27892	19176	14908
MR475	34151	25066	19217

© 2009 ACADEMY PUBLISHER

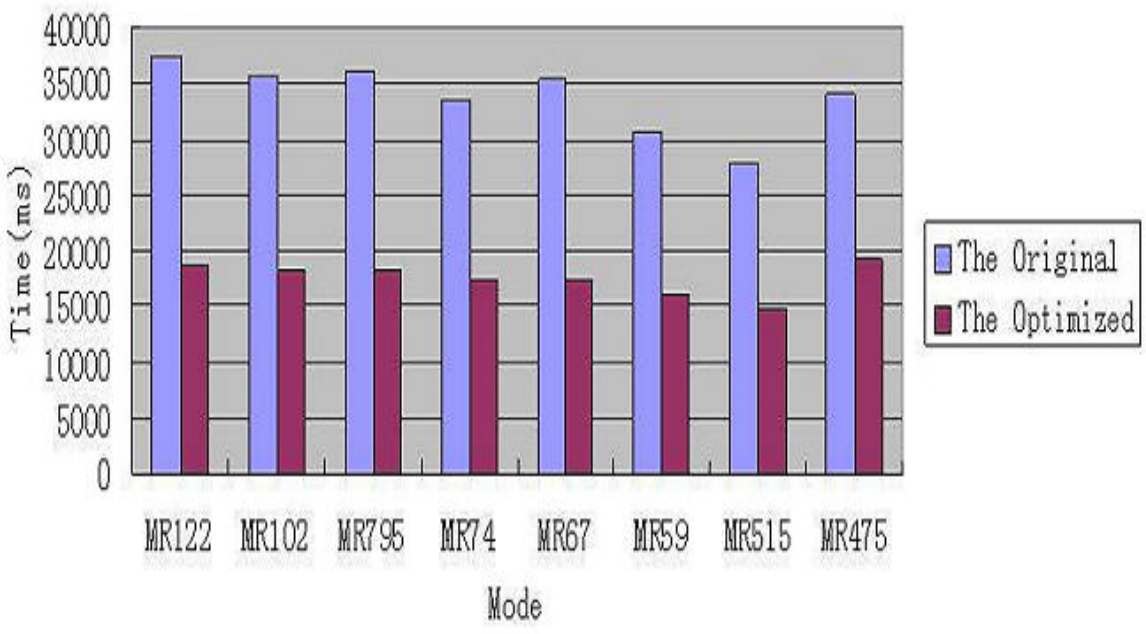

Fig. 3. The comparison of processing time of different modes

Sometimes, in the audio domain, the simple signal-to-quantization-noise ratio (SQNR) could not reflect the complexity of human hearing [12].For subjective test, we use MOS (Mean Opinion Score) to evaluate speech quality. A listener is required to evaluate the speech quality according the standard that 5 score represents excellent while 1 score stands for very annoying. Table Ⅳshows the result.

Fig. 4 shows the waveform and spectral view of the test speech, of which we can see litter difference. The original and the speech processed by eight modes are all presented.

The processing time of some modules is also measured. We take mode 12.2kb/s as the example. Since we have mainly modified the open-loop pitch analysis and codebook search parts, we focus on the time of these two functions. The running time is shown in Table. Ⅴ. The percentages of each part before and after the optimization are shown in Fig. 5.

TABLE Ⅳ
AVERAGE MOS SCORE

Modes	Score
MR122	3.95
MR102	3.88
MR795	3.62
MR74	3.57
MR67	3.52
MR59	3.40
MR515	3.20
MR475	3.04

TABLE Ⅴ
THE COMPARISION OF PROCESSING TIME OF FUCTIONS (12.2 kb/s)

Functions	Time (ms)	
	The original time	The optimized time
Open-loop pitch analysis	4931	512
Codebook search	12178	6091
Total	37293	18675

Ⅴ. CONCLUSION

This paper describes the implementation and optimization of the AMR speech coder on mobile device. Due to the high complexity and the constraint of computation resource of mobile device, it is a challenging work to achieve a real time codec on embedded processor. We have adopted some efficient algorithms to reduce the computational load. The instruction optimization has also been considered. The experiment result shows that nearly 50% of processing time is saved while maintained the speech quality. It is helpful to achieve high-efficiency when implementing multimedia application on the mobile devices.

ACKNOWLEDGMENT

We thank Tao Xia,Yi Gao, Weifang Han for their suggestions on structuring the paper.

© 2009 ACADEMY PUBLISHER

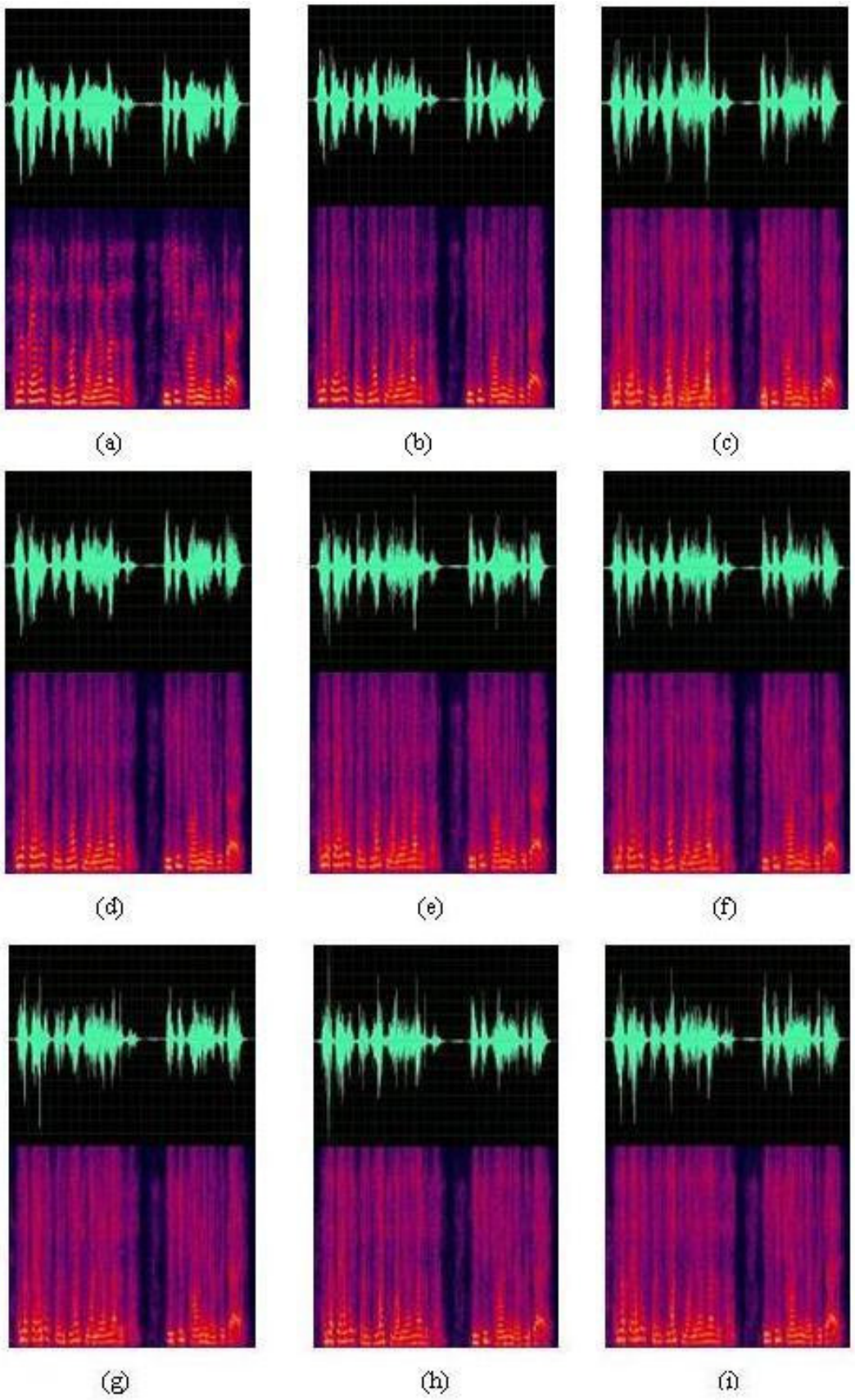

Fig. 4. The waveform and spectral view of speech
(a) The original; (b) 12.2kb/s; (c) 10.2kb/s; (d) 7.95kb/s; (e) 7.4kb/s; (f) 6.7kb/s; (g)5.9kb/s; (h) 5.15kb/s; (i) 4.75kb/s.

© 2009 ACADEMY PUBLISHER

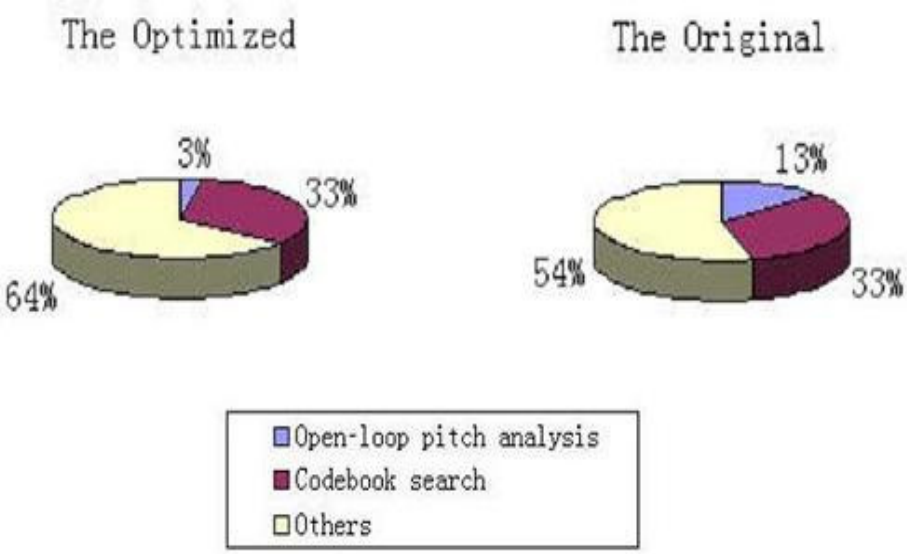

Fig. 5. Time rate of different parts

REFERENCES

[1] Sahafi L, Randhawa T.S, Hardy R.H.S. "Context-based complexity reduction of H.264 in video over wireless applications," *IEEE 6th Workshop on Multimedia Signal Processing*, Oct. 2004, pp.23 – 26,.

[2] Rao G.N, Prasad R.S.V, Chandra D.J, Narayanan S. "Real-Time Software Implementation of H.264 Baseline Profile Video Encoder for Mobile and Handheld Devices," *IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2006,vol 5, pp.14-19.

[3] Zhenyu Wei, Kai Lam Tang, Ngan, K.N. "Implementation of H.264 on Mobile Device," *IEEE Transactions on Consumer Electronics*. vol 53, issue 3, pp.1109-1116 ,Aug 2007.

[4] 3GPP TS 26.071, "AMR speech codec; general description."

[5] 3GPP TS 26.190, "AMR wideband speech codec transcoding functions".

[6] H J. Kim, D.G. Jee, M. H. Park, B. S. Yoon, and S. I. Choi, "The Real-Time Implementation of Multi-channel AMR Codec Using TMS320C62xx DSP", *Lecture Notes in Computer Science*, *Springer Berlin / Heidelberg*, Volume 2524/2003,pp 373-378,2003

[7] Andrew N Sloss. *ARM System Developer's Guide: Designing and Optimizing System Software*, Elsevier, 2005, pp.79-82.

[8] Goodacre J, Sloss A. N, "Parallelism and the ARM instruction set architecture", *IEEE Transactions on Computer*, vol 38, issue 7, pp. 42 – 50, July 2005.

[9] Kyung Jin Byun, Hee Bum Jung, Minsoo Hahn, Kyung Soo Kim. "Computationally efficient implementation of AMR speech coder," *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis*, Sept.2003. vol 1, pp.528 – 531.

[10] 3GPP TS 26.073, "ANSI-C code for the Adaptive Multi Rate (AMR) speech codec"

[11] [Online]. Available:http://msdn.microsoft.com

[12] R. Chamberlain, E. Hemmeter, R. Morley, and J. White. "Modeling the power consumption of audio signal processing computations using customized numerical representations". *Proceedings of Conference on the 36th Annual Simulation Symposium*. April 2002, pp.249-255.

Jie Yang was born in 1981. He received the B.S. degree from Huazhong University of Science and Technology (HUST) in 2004.He is currently pursuing the Ph.D. degree at HUST.

His main interests include signal processing, speech and video compression.

Shengsheng Yu was born in 1944, and received the B.E. degree in 1967.

He had been a visiting scholar in Germany from 1982 to 1983. He is currently a Professor at Huazhong University of Science and Technology. His main field of research includes computer network and storage, discrete signal processing and communication.

Jingli Zhou was born in 1946. She received the B.E. degree in 1969.

She visited USA from 1995 to 1996 as a scholar and has been honored of the State Department Special Allowance since 1999. She is a professor at Huazhong University of Science and Technology. Her main field of research includes computer network and multimedia signal processing.

© 2009 ACADEMY PUBLISHER