Processing Power Optimisation for PESQ

Jan Holub, Radislav Šmíd, Jindřich Očenášek

FEE CTU Prague, Dept. of Measurement K338, Technická 2, 166 27 Prague 6, Czech Republic

holubjan@feld.cvut.cz

Abstract

This article identifies several areas of possible reduction of processing power requirements of the PESQ algorithm (ITU-T P.862). As a result, PDA-based implementation of PESQ-based speech transmission quality measurement system for GSM networks is presented and required computational times are discussed.

Keywords

PESQ, processing power, PDA

1 Introduction

PESQ as recommended in ITU-T P.862 is world wide accepted algorithm that represents state of the art in the area of intrusive speech transmission quality measurements. However, it is well suited for powerful computing platforms, where even real-time performance in pipe-lined arrangement can be achieved (PESQ score is computed earlier than the acquisition of the consequent speech sample is finished). Problems arises in case of attempts of PESQ implementation to low-power devices like Personal Digital Assistant (PDA) or even GSM or UMTS mobile terminal. Due to many thousands of FFT operation required and due to usually poorly implemented floating-point arithmetic, the calculation may take even hours for 10s speech sample.

2 Work performed

2.1 Two Alternative Solutions

Two basic approaches have been studied in parallel:
a) Algorithm optimisation that keeps P.862 - compliant results
b) Algorithm simplification that does not keep compliance with P.862 but the correlation coefficient and maximum absolute errors between its MOS estimates and ACR MOS listening test results are comparable with that of PESQ.
This article deals with the case a) since the case b) has been followed in other contributions [2].

2.2 Optimisation Ways

Taking the code of PESQ delivered by ITU-T and compiling it into PDA using Visual C++ v 4.0 and PocketPC2003 SDK, we get the following processing times:

Sample	AMD Dutin 600MHz Calculation Time (min:sec)	XDA StrongARM 206MHz Calculation Time (hours:min:sec)
109	00:02,9	0:03:04
114	00:04,2	0:05:12
129	00:02,2	0:02:07
134	00:05,6	0:05:32
137	00:01,8	0:01:50
145	00:03,3	0:03:11
149	00:03,7	0:03:50
152	00:02,3	0:03:10
154	00:01,7	0:01:49
155	00:02,3	0:02:47
161	00:03,7	0:04:25
164	00:02,1	0:02:27
166	00:03,9	0:04:53
170	00:03,7	0:03:56
179	00:05,9	0:07:44
221	00:04,3	0:05:30
229	00:03,0	0:02:55
246	00:04,6	0:05:01
272	00:03,5	0:03:39

Table 1 Processing times for std speech samples for PC (AMD 600MHz) and PDA (XDA StrongARM 206MHz)

It is obvious that the calculation is not fast enough for practical applications since it is approximately 60 times slower than for PC. The following ways of optimisation have been identified:

-Optimal programming environment selection
-Pre-calculation of some parameters (applicable in case of limited reference speech sample set used during measurements)
-Efficient FFT implementation (4-base FFT)
-Replacement of FFT-based IRS filters by IIR-based filters

2.3 Pre-calculation of parameters

Certain parameters (gain compensation, IRS filtering) are calculated both for original and degraded samples. When known base of speech samples is used, it is possible to pre-calculate those parameters for original sample thus saving processing power and time later.

Algorithm part	Processing time [min:sec]
Gain compensation	1:00
IRS filters	1:00
Time alignment	1:00-5:00
Psychoacoustic model	0:40

Table 2 Approximate processing times for PESQ steps

By means of this procedure one half of the first and also of the second step of PESQ (see Table 2) can be pre-calculated, allowing around 1 minute time savings in entire duration.

2.4 Efficient FFT implementation

PESQ algorithm uses FFT algorithm of various length heavily. Number of FFT operations used in two examples are given in Tab. 3:

FFT	or109.wav	or179.wav dg179.wav
256	0	42
512	1731	18072
1024	180	255
2048	132	135
4096	0	0
8192	3	3
16384	0	0
32768	0	0
65536	0	0
131072	8	8
Total	**2054**	**18515**

Table 3 Number of FFT operations – examples

Therefore it is obvious that efficient implementation of FFT operation is crucial for calculation time. We have used 4-base FFT implemetation according to [5]. The results are given in Table 4:

Sample	Standard FFT Calculation Time (hours:min:sec)	4-base FFT Calculation Time (hours:min:sec)
109	0:03:04	0:01:38
114	0:05:12	0:02:30
129	0:02:07	0:01:18
134	0:05:32	0:02:54
137	0:01:50	0:01:03
145	0:03:11	0:01:39
149	0:03:50	0:02:00
152	0:03:10	0:01:34
154	0:01:49	0:01:05
155	0:02:47	0:01:30
161	0:04:25	0:02:16
164	0:02:27	0:01:18
166	0:04:53	0:02:28
170	0:03:56	0:02:04
179	0:07:44	0:03:38
221	0:05:30	0:02:50
229	0:02:55	0:01:33
246	0:05:01	0:02:47
272	0:03:39	0:02:01

Table 4 Improvement achieved by 4-base FFT deployment shown on std set of samples

2.5 Replacement of FFT-based IRS filters by IIR-based filters

We have also tested the possibility to replace FFT-based filters by IIR filtering that may provide, under certain circumstances, faster behaviour. Using the 10-th order elliptic filter insted of IRS FFT-based filter, we achieved around further 10% reduction of computational time in average. Due to the fact that the algorithm results differ slightly from PESQ scores, we have decided not to continue in this direction and FFT based filters have been kept in the implementation.

2.6 Optimal feature combination

As follows from the analysis above, as the optimal combination of features, efficient FFT and parameter pre-calculation appear. The results are listed in Table 5:

Sample	Original Implementation Calculation Time (h:min:sec)	4-base FFT plus DAT file Calculation Time (h:min:sec)
109	0:03:04	0:01:16
114	0:05:12	0:02:08
129	0:02:07	0:01:07
134	0:05:32	0:02:31
137	0:01:50	0:01:19
145	0:03:11	0:01:19
149	0:03:50	0:01:39
152	0:03:10	0:01:23
154	0:01:49	0:00:55
155	0:02:47	0:01:19
161	0:04:25	0:01:55
164	0:02:27	0:02:18
166	0:04:53	0:02:06
170	0:03:56	0:01:41
179	0:07:44	0:03:17
221	0:05:30	0:02:29
229	0:02:55	0:01:12
246	0:05:01	0:02:25
272	0:03:39	0:01:39

Table 4 Improvement achieved by 4-base FFT and DAT file deployment

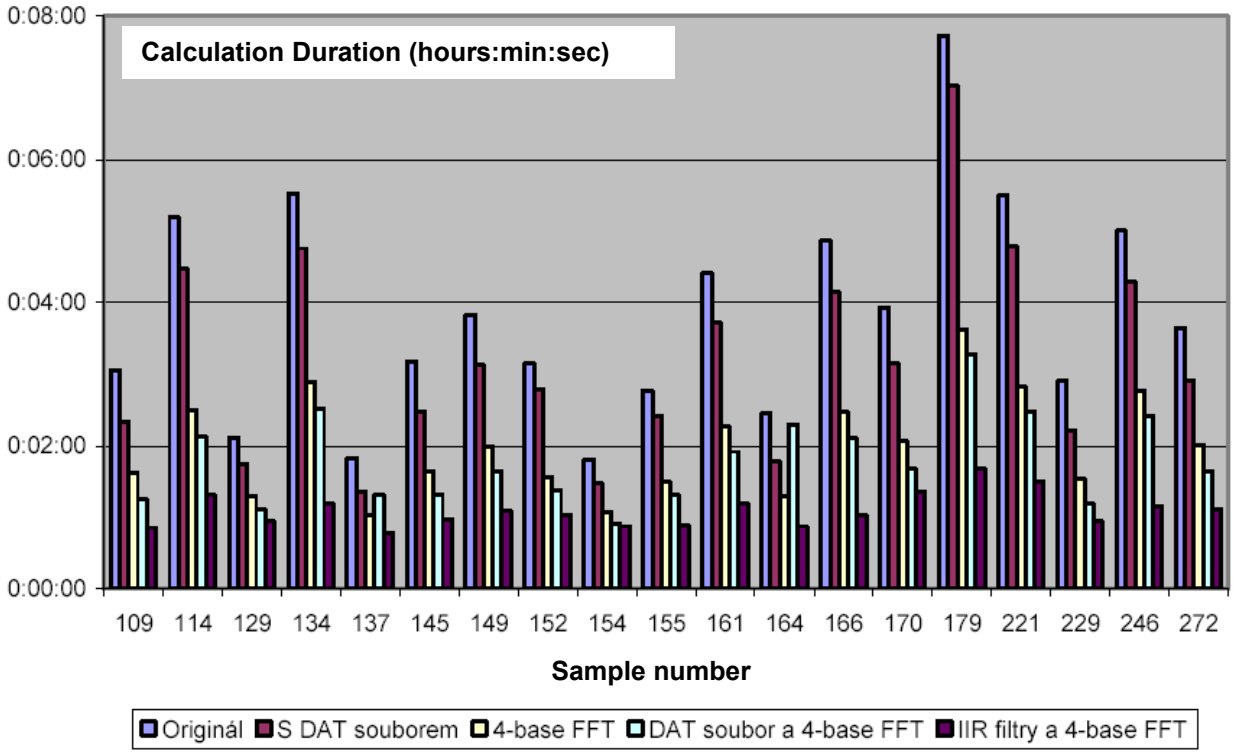

Fig. 1 Calculation duration for std speech sample set

The graphical representation of all the results discussed before is available at Fig. 1. Also IIR filtering is included for completeness, however, the resulting algorithm cannot be considered as P.862-compliant in this case.

Fig. 2 shows setup screen for PDA-based resulting application and the result presentation screen is depicted in Fig. 3

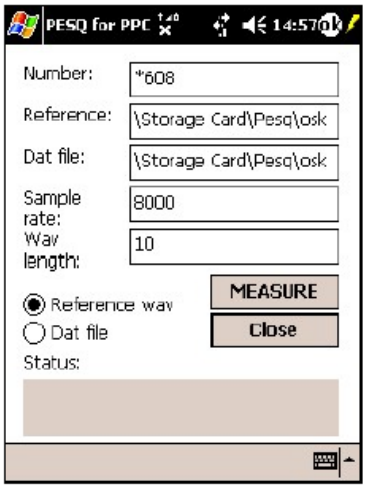

Fig. 2 Screenshot "Settings"

Fig. 3 Screenshot "Results"

3 Conclusion

A P.862-compliant version of PESQ algorithm, optimised for PDA environment with reduced computational and memory capabilities has been developed. For 10s speech sample, it achieves the average computational time of 40s. Real-time versions (processing time less than sample duration) are available, too, however, the MOS estimates differs

slightly from that of PESQ. The picture, showing PDA equipped with PESQ capability as a part of speech transmission quality monitoring system, is shown in Fig. 4.

Fig. 4 PDA-based application

References

[1] ITU-T P.862, Perceptual Evaluation of Speech Quality, ITU-T, February 2001

[2] Dresler, T., Holub, J., Šmíd, R.: Voice Transmission Quality Measurement based on Wavelet Transform, XVII IMEKO World Congress, June 22-27, 2003, Dubrovnik, Croatia

[3] Frigo M., Johnson S.G., Fastest Fourier Transform in the West [online] <http://www.fftw.org>

[4] Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P. Numerical Recipes in C, The Art of Scientific Computing - Second Edition. Cambridge: Cambridge University Press, 2002

[5] Ooura T., Ooura's Mathematical Software Packages [online].< http://momonga.t.u-tokyo.ac.jp/~ooura/>