2001년도 대한전기학회 하계학술대회 논문집 2001.7.18-20

LPC 계수의 최적 양자화에 기초한 음성 코더 구현

이우종, 박지태, 장태규
중앙대학교 전자전기공학부

Implementation of a CELP coder based on optimum quantization of the LPC coefficients

W.J Lee, J.T Park, T.G Chang
School of Electrical Engineering Chung-Ang University

Abstract - The quantization of the LPC parameters is a very important aspect of the speech compression algorithm. This paper analyzes the quantization effect of the LPC coefficients and presents the implementation of a fixed-point CELP coder based on the LPC analysis.

1. 서 론

Linear Predictive Coding(LPC)은 저비트율 음성 압축 앨고리즘에서 가장 유용한 방법이며 음성 합성에서 매우 중요한 역할을 한다. LPC는 음성의 주파수 모양을 정확히 표현할 수 있으며 연산량 부담이 적어서 널리 사용되고 있다. 일반적인 CELP에서는 효과적인 코딩을 위하여 LPC 계수를 LSP로 변환하여 압축하는 기법을 사용한다. 이처럼 LPC 계수는 음성을 분석하고 합성하는 방법에 있어 매우 중요한 의미를 가지고 있다. 본 논문에서는 LPC 계수의 최적 양자화에 관한 연구와 더불어 MPEG-4 CELP 코더를 고정소수점 연산구조로 구현하였다. 이를 위해 정밀도(Precision)와 동적 범위(Dynamic Range)를 고려하여 스케일링 기법을 사용하였으며 수학 함수들은 미리 계산하여 테이블링 기법을 적용하였다. 인지(Perception) 테스트를 통하여 부동소수점 코더와 음질의 열화 상태를 비교하였다.

2. 본 론

2.1 LPC Analysis

2.1.1 Basic Principles of LPC

일반적인 음성코더에서 LPC 계수를 구하기 위하여 사용하는 방법은 Auto-correlation Method이다. 이는 각 서브프레임에 대해 Hamming Windowing을 적용하고 auto-correlation을 계산한다. 그리고 이 값을 이용하여 Levinson-Durbin 앨고리즘을 통해 LPC 계수를 구한다. Levinson-Durbin 앨고리즘은 다음과 같다.

$$E_n^{(0)} = R_n^{(0)} \tag{1}$$

$$k_i = [R_n(i) - \sum_{j=1}^{i-1} a_j^{i-1} R_n(i-j)]/E_n^{(i-1)}, \quad 1 \le i \le p \tag{2}$$

$$a_i^{(i)} = k_i \tag{3}$$

$$a_j^{(i)} = a_j^{(i-1)} - k_i a_{i-j}^{(i-1)}, \quad 1 \le j \le i-1 \tag{4}$$

$$E_n^{(i)} = (1-k_i^2)E_n^{(i-1)} \tag{5}$$

이렇게 구한 LPC 계수는 인지가중(Perceptual Weighting) 필터를 구성하는데 사용되며 LPC 합성(Synthesis)필터는 다음과 같다.

$$H(z) = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{6}$$

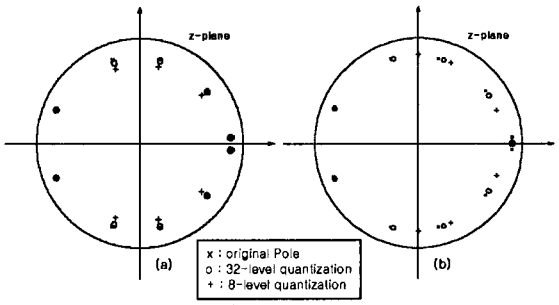

그림 1 LPC 합성필터 pole의 반지름과 위상 양자화
(a) 반지름을 32-level과 8-level로 양자화.
(b)위상을 32-level과 8-level로 양자화.

LPC 합성(Synthesis) 필터와 인지가중(Perceptual Weighting) 필터에 사용되는 LPC 계수는 스펙트럼의 왜곡 없이 적은 비트로 양자화 하는 것이 주된 관심사항이라 할 수 있다. 저비트율 음성 코더의 경우 이를 위해 LPC 파라미터를 LSP(Line Spectral Pairs)로 변환하여 양자화 하는 방법을 사용한다.

2.1.2 Line Spectrum Pairs(LSP)

음성 합성을 위한 디지틀 필터는 선형 예측 분석으로부터 구해지며 다음과 같이 주어진다.

$$H(z) = \frac{1}{A(z)} \tag{7}$$

$$A(z) = 1 - \sum_{k=1}^{p} a_k z^{-k} \tag{8}$$

LPC 파라미터의 변환 형태인 Line Spectrum Pairs(LSP)는 반사계수(Reflection coefficients)보다 저비트율을 제공하고 동등한 음질을 제공한다. LSP 변환은 A(z)의 zeros(또는 LPC 합성필터의 poles)들을 두 개의 다항식인 P(z)와 Q(z)를 통해 단위 원(unit circle) 위로 이동시킨다.

$$P(z) = A(z) + Z^{-(p+1)}A(z^{-1}) \tag{9}$$

$$Q(z) = A(z) - Z^{-(p+1)}A(z^{-1}) \tag{10}$$

$$A(z) = [P(z) + Q(z)]/2 \tag{11}$$

P(z)와 Q(z)는 손실 성도(vocal tract) 응답인 A(z)의 무손실 모델에 대응한다. P(z)와 Q(z)의 zeros는 주파수가 단위 원을 따라 증가할 때 교대로 존재한다. 또한 LSP 계수는 A(z)의 각 복소수 zero가 각 P(z)와 Q(z)중에 한 개의 zero로 대응되기 때문에 포르만트(Formant) 주파수 해석을 가능하도록 한다.

그림 1은 z-plane상에서 반지름과 위상에 대해 양자화를 실시한 결과이다. 그림 (a)는 위상은 유지시키고 반지름을 32-level, 8-level로 각각 양자화 하였고, 그림 (b)는 반지름은 유지시키고 위상을 32-level, 8-level로 양자화 하였다. 이 실험을 통하여 LPC 합성(Synthesis) 필터의 pole은 반지름의 변화보다 위상의

변화가 귀의 특성에 영향을 많이 미치는 것을 확인하였다.

2.2 MPEG-4 CELP coder의 고정소수점 연산 구조 구현

2.2.1 고정 소수점 음성코더 개발 과정

그림 2는 고정 소수점 기반의 음성코더 개발 과정을 나타내었다. 일반적으로 음성 코더는 초기에 부동 소수점 연산구조에 기반하여 개발되고 검증을 실시한 후 DSP 어셈블리로 변환하기 전에 중간 과정인 고정소수점 구조로의 변환이 필수적이다. 고정소수점 연산구조로의 변환과정에서 DSP 프로세서가 제공하지 않는 수학함수들은 테이블 기법을 적용하여야 하며, 음성 모델 파라미터에 대해 scale factor를 미리 정의하여야 한다. 각 모듈별로 고정 소수점 연산 구조로 변환하고 이를 통합하여 인지(perception) 테스트를 실시하였다.

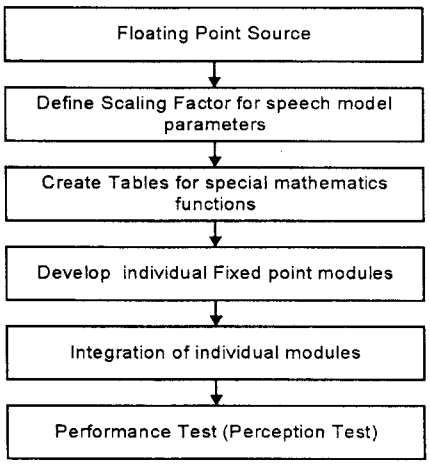

그림 2 고정 소수점 음성코더의 개발 차트

2.2.2 음성 파라미터 스케일링

고정 소수점 DSP는 제한된 dynamic range내에서 데이터를 다루기 때문에 음성 파라미터에 대한 scale factor들이 미리 정의 되어야 한다. 적절한 스케일링은 각 파라미터의 dynamic range를 극대화 시켜주며 라운딩 에러를 최소화 시켜준다. 또한 overflow나 saturation을 방지해 준다.

2.2.3 수학 함수들에 대한 테이블링 적용기법

표준 DSP 명령어가 제공하지 않는 수학함수들은 미리 계산하여 테이블링 하는 것이 필수적이다. 따라서 구현할 코더에서 사용되는 cosine 함수와 log(1+x) 함수, 10^x 함수를 테이블링 하였다. cosine 함수는 512개로 테이블링 하였고, log(1+x) 함수는 [1 ~ 600]의 입력범위에 대해 600 엔트리를 가지도록 하였으며 결과값의 범위가 [0.3 ~ 2.7789]이기 때문에 4로 스케일링하여 사용하였다. 10^x 함수는 입력범위가 [1.0 ~ 3.0]이고 이를 384 엔트리를 가지도록 테이블링 하였다.

2.2.4 정규화(Normalization)

고정소수점 연산구조에서 가장 주의해야 할 사항은 에러의 축적이다. 이의 해결방법 중의 한가지가 입력 데이터의 정규화이다. 이것은 곱하기 연산이나 나누기 연산에서 정밀도(Precision)를 극대화 해준다. 정규화 명령어는 일반적으로 DSP 명령어에 포함되어 있으며 MSB (most significant bit)의 zero가 없어질 때까지 왼쪽으로 쉬프트 한다.

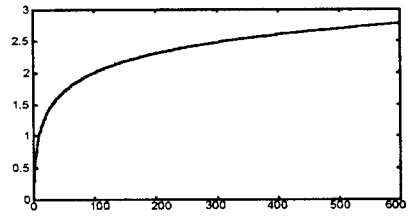

그림 3 log(1+x)의 입출력 범위

2.2.5 모듈안에서의 파라미터 스케일링

고정 소수점 연산구조를 구현하기 전에 모든 Scale Factor들을 미리 정의하는 것이 필요하다. 또한 부동소수점 모듈을 통하여 Dynamic range의 계산이 필요하다. 표 1은 부동 소수점으로 구현되어 있는 합성(Synthesis) 필터이고 이를 고정 소수점 연산구조로 변환시킨 것이 표 2이다. 표 2에서는 4로 스케일된 LPC 계수를 입력으로 받아서 결과는 원래의 스케일로 복원하여 내준다.

```
for (j=0; j < n; j++) {
    s = (float)0.0;
    for (i=0; i < np; i++)
        s = s - a[i] * pm[i];
    xr[j] = di[j] + s;
    for (i=2; i < np+1; i++)
        pm[np-i+1] = pm[np-i];
    pm[0] = xr[j];
}
```

표 1 부동소수점 합성(Synthesis) 필터

```
for (j=0; j < n; j++) {
  Fixs = (long)0;
  for (i=0; i < np; i++)
      Fixs=L_SUB(Fixs,L_MULT(Fixa[i],
                  Fixpm[i]));
  xr[j] = di[j] + s;
  Fixxr[j]=ADD(Fixdi[j], ROUND(L_SHL(Fixs,2)))
  for (i=2; i < np+1; i++)
      Fixpm[np-i+1] = Fixpm[np-i];
  Fixpm[0] = Fixxr[j];
}
```

표 2 고정소수점 합성(Synthesis) 필터의 스케일링

만일 중간 결과가 범위 내에서 값이 너무 크다면 중간 스케일링(Intermediate Scaling)이 요구된다. 곱하고 축적(multiply and accumulate)하는 연산에 대해 overflow나 saturation을 방지하기 위하여 중간에 적당한 값으로 스케일 해준다. 표 3은 Levinson-Durbin recursion을 구하기 위하여 Autocorrelation 계수를 구하는 모듈의 일부분이다. 이것은 두 입력을 곱하고 축적하는 전형적인 구조이다. 이를 고정 소수점으로 변환할 때 표 4에서와 같이 두 입력을 16 bit으로 받아서 곱한 값을 32 bit으로 저장하고 오버플로우를 방지하기 위하여 1 bit을 내려서 축적한다. 이렇게 축적된 값은 다시 16 bit으로 반올림하여 결과값으로 내준다.

```
for (l = 0; l < N - i; l++)
    temp += *pin1++ * *pin2++;
*acf++ = temp;
```

표 3 부동 소수점 축적 연산구조

```
for (l = 0; l < N - i; l++) {
    Acc1 = L_MULT(*pin1++ , *pin2++);
    Acc1 = L_SHR(Acc1,1);
    Acc0 = L_ADD(Acc0,Acc1);
}
*Fixacf++ = ROUND(Acc0);
```

표 4 고정 소수점 축적 연산구조의 중간 스케일링

2.2.6 LSP 계수를 위한 앨고리즘 변환

부동 소수점 연산 기반의 코더에서는 LPC 계수를 LSP 계수로 변환하기 위하여 Chebyshev series method를 이용하였다. 다음의 식을 Chebyshev 다항식을 이용하여 확장하고 x=cos(w)에 대입하면 실수 구간 [-1 1] 사이에 근이 존재하게 된다.

$$P(z) = 2e^{ipw/2}[A_0\cos(\frac{p}{2}w)+A_1\cos(\frac{p-2}{2}w) + \ldots+\frac{1}{2}A_{p/2}] \quad (12)$$

$$Q(z) = 2e^{ipw/2}[B_0\cos(\frac{p}{2}w)+B_1\cos(\frac{p-2}{2}w) + \ldots+\frac{1}{2}B_{p/2}] \quad (13)$$

이 방법을 고정 소수점 연산으로 구현하면 오차가 Real root method 방법으로 구현한 것보다 크고 사인함수, 코사인 함수 그리고 역 코사인 함수 총 3개의 함수에 대해 테이블링이 요구된다. 이 3개의 테이블링에 의해 오차가 누적되는 결과를 초래한다. 이에 반해 Real root method 방법은 (12),(13)식에 x=cos(w)를 대입하여 식을 정리하면 아래의 식을 얻을 수 있고

$$P_{10}(x) = 16A_0x^5+8A_1x^4+(4A_2-20A_0)x^3 +(2A_3-8A_1)x^{2+(5A_0}-3A_2+A_4)x +(A_1-A_3+0.5A_5) \quad (14)$$

$$Q_{10}(x) = 16B_0x^5+8B_1x^4+(4B_2-20B_0)x^3 +(2B_3-8B_1)x^{2+(5B_0}-3B_2+B_4)x +(B_1-B_3+0.5B_5) \quad (15)$$

근은 아래의 식에 의해 구할 수 있다.

$$LSF(i) = \frac{\cos^{-1}(x_i)}{2\pi T}, \quad \text{for } 1\le i\le p \quad (16)$$

Real root method는 코사인 테이블 한 개로 구현이 가능하여 오차측면과 메모리 측면에서 Chebyshev series method에 비해 성능이 우수함을 확인하였다.

2.2.7 성능 테스트

음성 코더의 성능 실험은 주로 객관적인 테스트(objective test)와 주관적인 테스트(subjective test)로 실시된다. 본 연구에서는 주관적인 테스트인 인지(perception) 테스트를 실시하였다. 시료는 남자 음성 5문장과 여자 음성 5문장을 사용하였다. 이를 부동 소수점 기반의 코더와 고정 소수점 기반으로 구현한 코더를 가지고 원본과의 음질 열화를 비교하였다. 그림 4는 여자 음성을 시료로 하여 부동 소수점 기반의 코더와 본 연구에서 구현한 고정 소수점 기반의 코더를 통과한 음성 파형을 출력한 것이다. 그림(a)는 여성 화자의 음성 파형이고 (b)는 부동 소수점 기반의 코더를 통과한 음성 파형이고 (c)는 고정 소수점 기반의 코더를 통과한 음성 파형이다. 인지 테스트 결과 대부분의 시료에서 고정 소수점 기반의 코더를 통과한 음성은 부동 소수점 코더를 통과한 음성과 비교하여 음질의 열화가 없음을 확인하였다.

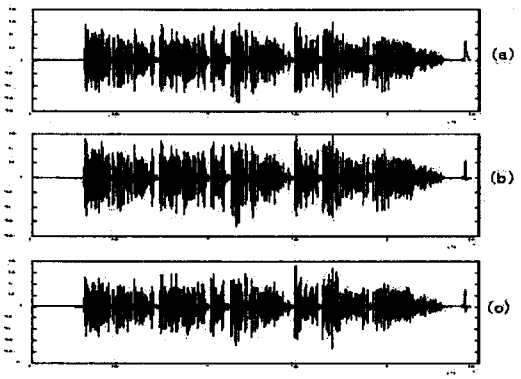

그림 4 MPEG-4 CELP coder를 통과한 음성파형 (a) 원본 음성, (b) 부동 소수점 코더를 통과한 음성파형, (c) 고정 소수점 코더를 통과한 음성파형

3. 결 론

본 논문에서는 저비트율 음성 압축 앨고리즘에서 많이 사용되는 Linear Predictive Coding(LPC)과 그것의 양자화 방법으로 널리 사용되는 Line Spectrum Pairs(LSP)에 관하여 살펴보았다. 또한 LPC 계수로 구성되는 LPC 합성(Synthesis) 필터의 pole 특성에 관하여 연구하였다. 또한 부동 소수점 기반의 음성 압축 앨고리즘인 MPEG-4 CELP coder를 DSP 어셈블리로 구현이 가능하도록 고정 소수점 기반으로 구현하였다. 고정 소수점 연산에서 고려해야 할 정밀도(Precision)와 동적범위(Dynamic Range)를 보장하기 위하여 정규화(Normalization)와 스케일링 방법을 사용하였다. 그리고 DSP 명령어가 지원하지 않는 수학 함수들은 미리 계산하여 테이블링 기법을 적용하였다. 인지 테스트 결과 부동 소수점 코더와 비교하여 대부분의 시료에서 음질의 열화가 없음을 확인하였다. 그러나 10개의 시료 중에 일부는 부동소수점 코더와 비교하여 음질의 열화가 약간 발생하는 것도 확인하였다. 앞으로 앨고리즘의 개선과 코드, 테이블, 스케일링의 최적화에 관한 연구가 수행될 필요가 있다.

[참 고 문 헌]

[1] ISO/IEC 14496-3 Part 3: Audio Subpart 3 : CELP

[2] A.M.Kondoz, *Digital Speech*, John Wiley & Sons, 1994

[3] Ludy Liu, "Fixed Point Vocoder Implementation," *IEEE Circuits and Systems, Proceedings of the 40th Midwest Symposium on*, pp. 710 -715 vol.2. 1998

[4] L.R.Rabiner, R.W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, 1978

[5] Peter Kabal, Ravi, "The computation of Line Spectral Frequencies Using Chebyshev Polynomials," *IEEE Trans on ASSP*, pp. 1419-1425, December 1986.

[6] Douglas O'Shaughnessy, Speech Communications, IEEE Press, 2000