論文2003-40SP-4-8

펄스위치 재검색 방법을 이용한 AMR-WB 여기 코드북 검색

(AMR-WB Algebraic Codebook Search Method Using the Re-examination of Pulses Position)

許 焟 *, 李 寅 誠 **, 池 德 求 ***, 尹 炳 植 ***, 崔 松 仁 ***

(Seok Hur, Insung Lee, Deockgu Jee, Byung-Sik Yoon, and Song-In Choi)

요 약

3GPP에서 광대역 음성부호화 알고리즘으로 채택한 AMR-WB 부호화기는 8개의 모드로 동작되고 전송률을 23.85 kbps 까지 높임에 따라 음성부호화기 프로그램 및 데이터 메모리 크기가 매우 크게 증가되었으며, 합성필터의 입력신호인 고정 코드북의 펄스 수가 기존의 협대역 AMR 부호화기 보다 5 배 이상 증가함으로써 코드북 검색에 많은 계산량을 요구하고 있다. AMR-WB는 적당한 계산량에서 비교적 좋은 음질을 유지하기 위한 방법으로 depth-first 트리 검색 방법을 사용하고 있다. 본 논문에서는 여기코드북 검색 시 레퍼런스 벡터를 중심으로 적당한 계산량을 갖는 검색 방법으로 검색시 기준이 되는 레퍼런스 벡터를 좀 더 정확하게 예측하여 성능 향상을 하였으며, 펄스위치 재검색 알고리즘을 적용하여 depth-first 트리 검색 방법의 문제점을 개선시켜 1/2 이상의 계산량을 줄이면서 같은 성능을 유지하는 AMR-WB 여기코드북 검색기를 설계하였다.

Abstract

We propose a new method to reduce the complexity of excitation codebook search. The preselected excitation pulses by the coarse search method can be updated to pulses with higher quality performance measure. The excitation pulses can arbitrarily be deleted and inserted among the searched pulses until the overall performance achieves. If we use this excitation pulse search method in AMR-WB, the complexity required for excitation codebook search can be reduced to half the original method while the output speech maintains equal speech quality to a conventional method.

Keyword : AMR-WB, ACELP, Speech Coder, Codebook Search Algorithm

* 正會員, 忠北大學校 電波工學科, 컴퓨터情報通信研究所
(Dept. of Radio Science & Engineering Chungbuk National University)
** 正會員, 忠北大學校 電氣電子컴퓨터工學部, 컴퓨터情報通信研究所
(School of Electrical & Computer Engineering Chungbuk National University)
*** 正會員, 韓國電子通信研究員 移動通信研究所
(Mobile Telecommunication Research Laboratory Electronics and Telecommunications Research Institute)
接受日字:2003年2月26日, 수정완료일:2003年6月26日

Ⅰ. 서 론

2002년부터 새롭게 선보인 IMT-2000 이동통신 서비스의 가장 큰 특징은 넓은 대역폭(2Mbps)을 사용하며 음성에서 동영상까지 다양한 멀티미디어 데이터를 전송할 수 있다는 것이다. 음성통신에 있어서도 기존의 협대역 음성신호 뿐만 아니라 광대역 음성 통신도 가능하여진다[1, 2]. 또한 최근에 활발하게 보급되고 있는 고속 인터넷망에서는 전송 대역폭의 확대로 음질이 좋

은 광대역 음성과 오디오 전송도 실시간 전송이 가능하게 되었다.

현재의 광대역 음성부호화 방법 표준화는 1986년 파형 부호화 방식의 64 kbps Subband ADPCM 부호화기가 ITU-T G.722 표준안으로 사용되고 있고 1999년에는 DCT 변환을 이용한 32 kbps 광대역 음성 부호화 방법(G.722.1)이 표준화되었다[3, 4]. 인터넷, 이동통신망에서의 저전송률 광대역 음성부호화 방법의 필요성으로 인해 24kbps 이하에서 광대역 음성부호화 방법의 표준화 작업이 2001년 3GPP에서 이루어졌으며 협대역 AMR(Adaptive Multi rate)부호화기와 같은 구조의 AMR-WB 방식을 채택하였다[5]. 지금까지의 광대역 음성부호화기는 24kbps에서 64kbps 전송률에서 서브밴드 ADPCM 방식이나, DCT 변환으로 양자화하는 방법의 주파수 영역 파형 부호화 방법을 사용하고 있으며, 16kbps~24kbps 전송률에서는 분석/합성 구조인 CELP (Code Excited Linear Prediction)방식을 주로 사용한다. 3GPP에서 광대역 음성부호화 알고리즘으로 채택한 AMR-WB 부호화기는 8개의 모드로 동작되고 전송률을 23.85 kbps 까지 높임에 따라 음성부호화기 프로그램 및 데이터 메모리 크기가 매우 크게 증가되었으며, 합성필터의 입력신호인 고정 코드북의 펄스 수가 기존의 협대역 AMR 부호화기 보다 5배 이상 증가함으로써 코드북 검색에 많은 계산량을 요구하고 있다.

AMR-WB는 적당한 계산량에서 비교적 좋은 음질을 유지하기 위한 방법으로 depth-first 트리 검색 방법을 사용하고 있다. 기존의 CELP방식에서의 여기 코드북 검색시 모든 펄스 조합을 검색하는 방법으로 계산하기에는 펄스의 수가 많아 계산량을 줄이고자 depth- first 트리 검색 방법을 사용한다[6-8].

본 논문에서는 CELP방식 여기코드북 검색의 계산량을 줄이고 성능은 유지하는 펄스 검색 알고리즘을 제안한다. 2장에서 AMR-WB의 여기코드북 검색 방법인 depth-first 트리 검색을 살펴보고, 3장에서 레퍼런스 벡터의 정확한 예측과 위치 검색에 기반을 둔 반복 검색 알고리즘을 제안한다. 4장에서 3장의 알고리즘을 조합해 AMR-WB에 적용해 기존의 검색보다 적은 계산량으로 유사한 성능을 얻을 수 있는 방법을 살펴본다. 5장에서 NTT의 음성으로 시뮬레이션을 수행한 결과를 살펴보고, 마지막으로 결론에서 시뮬레이션 결과를 통해 제안된 알고리즘의 타당성을 보이도록 하겠다.

Ⅱ. AMR-WB의 여기코드북 검색

AMR-WB는 320개의 16비트 uniform PCM 포맷의 16000samples/s의 샘플링률의 데이터를 한 블럭으로 입력을 받아 132, 177, 253, 285, 317, 365, 397, 461, 477 비트로 인코딩해 출력으로 내보내고 이를 받아 디코딩 과정을 거쳐 다시 320개의 음성 샘플을 만들어 낸다. ACELP(Algebraic CELP)를 기반으로 한 광대역 음성부호화 알고리즘의 다중 모드 ACELP를 AMR-WB라고 부른다.

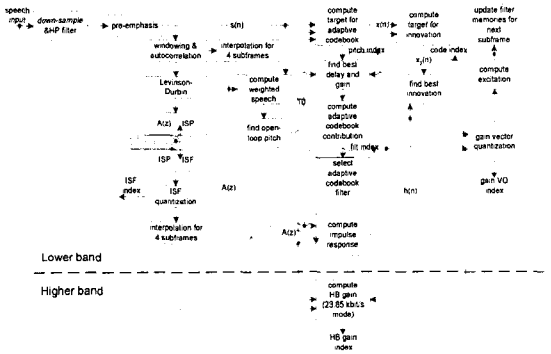


그림 1. ACELP의 간략화된 블록도
Fig. 1. Block diagram of the ACELP encoder.

부호화기의 신호의 흐름은 <그림 1>과 같다. LP 파라미터들은 ISP(Immittance Spectrum Pairs)로 변환되고, S-MSVQ(Split-MultiStage Vector Quanti- zation) 방식으로 벡터 양자화한다. 음성프레임은 각각 5ms의 4개의 부프레임으로 나뉘며 적응 및 고정 코드북 파라미터들은 매 부프레임마다 전송된다. 양자화되기 전의 LP 파라미터 또는 서로 삽입된 변형들은 부프레임에 의존하여 사용된다. 개방루프 피치 지연은 지각 가중된 음성신호에 근거한 각각의 부프레임 또는 프레임별로 예측값을 한번 구한다.

2.1 여기코드북 검색

목표신호 $x(n)$은 적응 코드북 기여를 제거함으로써 업데이트 되고, 이 새로운 목표신호 $x_2(n)$은 여기 코드북 검색에 사용된다.

여기 코드북은 가중 입력 음성과 가중 합성 음성사이의 평균 자승 오차를 최소화하는 것에 의해서 검색된다. 폐구간 피치 검색에 사용된 표적 신호는 적응코

드북 기여를 감산하는 것에 의해서 갱신된다.

$$x_2(n) = x(n) - g_p y(n) \quad n = 0, \ldots, 63 \tag{1}$$

필터링된 적응 코드북 벡터는 $y(n) = v(n) * h(n)$이고 양자화 되지 않은 적응 코드북 이득은 g_p이다.

행렬 $\boldsymbol{H}$는 대각성분 $h(0), h(1), \ldots, h(63)$을 가진 하위 토플리쯔 컨볼루션 행렬로 정의하고, $\boldsymbol{d} = \boldsymbol{H}^t \boldsymbol{x}_2$는 표적신호 $x_2(n)$과 임펄스 응답 $h(n)$의 상관도이고, $\boldsymbol{\Phi} = \boldsymbol{H}^t \boldsymbol{H}$는 $h(n)$ 상관행렬이다.

$$d(n) = \sum_{i=n}^{63} x_2(i) h(i-n), \quad n = 0, \ldots, 63 \tag{2}$$

$$\Phi(i,j) = \sum_{n=j}^{63} h(n-i) h(n-j), \quad i = 0, \cdots, 63, \quad j = i, \cdots, 63 \tag{3}$$

여기 코드북은 다음 검색 기준을 최대화하는 펄스 위치들을 선택한다.

$$Q_k = \frac{(\boldsymbol{x}_2^t \boldsymbol{H} \boldsymbol{c}_k)^2}{\boldsymbol{c}_k^t \boldsymbol{H}^t \boldsymbol{H} \boldsymbol{c}_k} = \frac{(\boldsymbol{d}^t \boldsymbol{c}_k)^2}{\boldsymbol{c}_k^t \boldsymbol{\Phi} \boldsymbol{c}_k} = \frac{(R_k)^2}{E_k} \tag{4}$$

검색의 목적은 위에 기술한 방법을 통해 최적화된 펄스 위치를 가지고 코드벡터를 결정하는 것이다. 위에서 언급했던 Q_k를 최대화하는 것을 기본적으로 선택을 하게 된다. <그림 2>는 depth-first 트리 검색 구조를 나타낸 것이다.

2.2 Depth-first 트리 검색 방법

검색 복잡도를 줄이기 위해 빠른 검색 방법인 depth-first 트리 검색 방법을 사용 한다. 이 방법은 한 번에 N_m개의 펄스를 결정하게 된다. 다음과 같이 N_p의 가능성 있는 펄스가 M개의 비어있지 않은 subset으로 나누어지게 된다.

$$N_1 + N_2 + \cdots + N_m + \cdots + N_M = N_p \tag{5}$$

AMR-WB에서는 한번에 두 개의 펄스를 검색하는 특별한 형태의 depth-first 트리 검색 방법이 사용되는데, 이는 $N_m = 2$를 의미하며 두 펄스는 인접한 트랙에서 검색된다. 64×64=4k워드의 메모리 공간을 요구하는 매트릭스 Φ를 사용하지 않고 메모리 요구량을 줄이는 효과적인 접근 방법이 사용된다. 이런 접근 방법에서 검색 절차는 미리 계산되어 저장된 상관 매트릭스의 필요한 부분들만 선택하여 수행된다. 이 부분은 행렬 Φ의 대각 원소에 부합하는 상관도뿐만 아니라 연속적인 트랙들에서 가능한 펄스 위치들에 부합한 임펄스 응답의 상관도와 연관된다.

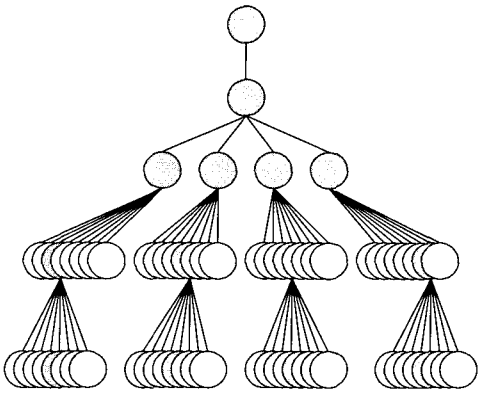


그림 2. Depth-first 트리 검색 구조도
Fig. 2. Depth-first tree Search Architecture.

두 개의 가능한 펄스 조합을 테스트하는 데 있어 복잡도를 감소시키기 위해서, 테스트되는 첫 번째 펄스의 위치를 제한된 수로 고정시킨다. 많은 수의 펄스일 경우에 검색 트리에서 상위 레벨에서 몇 개의 펄스들은 고정 시킨다. <그림 3>에서처럼 처음 몇 개의 펄스를 고정하고 다음 펄스들을 검색한다. 비트율에 따라 첫 번째 혹은 두 번째 까지의 고정된 펄스들의 위치를 좀 더 효과적으로 고려하기 위해서, 음성 신호 관련된 "pulse-position likelihood-estimate vector" b를 사용한다. 이 레퍼런스 벡터 b의 p번째 원소 $b(p)$는 검색하고자 하는 최적의 코드벡터에서 위치 p를 점유한 펄스의 확률 특성을 나타낸다. 처음 i번째 펄스 사인을 검색을 하지 않고 레퍼런스 벡터의 해당 펄스로 사용하며, 이를 "signal-selected pulse amplitude"라고 한다. AMR-WB에서 사용하는 레퍼런스 벡터 $b(n)$은 다음과 같다.

$$b(n) = \sqrt{\frac{E_d}{E_r}}\, r_{LTP}(n) + \alpha\, d(n) \tag{6}$$

$E_d = \boldsymbol{d}^t \boldsymbol{d}$는 $d(n)$의 에너지, $E_r = \boldsymbol{r}_{LPT}^t \boldsymbol{r}_{LTP}$ 는 장구간 예측의 잔여 신호인 $r_{LTP}(n)$의 에너지이다. 스케일링 배율 α는 비트율이 증가되면 낮아진다. 6.6과 8.58 모드에서 $\alpha = 2$이고, 12.65, 14.25, 15.85 모드에서 $\alpha = 1$, 18.25 모드는 $\alpha = 0.8$, 19.85 모드에서는 0.75, 그리고 23.05 모드와 23.85모드에서는 $\alpha = 0.5$이다.

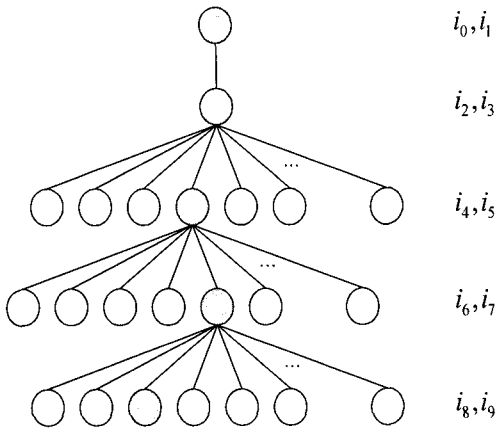


그림 3. Depth-first 트리 검색 순서
Fig. 3. Depth-first tree Search Sequence.

모든 모드에서의 검색 구조는 유사하다. 한번에 두 개의 펄스가 검색되고, 두 개의 펄스는 인접한 트랙에 위치하게 된다. 이는 검색된 펄스가 각각 트랙 T0-T1, T1-T2, T2-T3, T3-T0에 위치하는 것을 의미한다. 그리고 수정된 신호 $d'(n)$이 미리 결정된 sign을 포함해 계산된다.

각 트리 레벨에서 $R_{hv}(m)$값은 두 개의 트랙에서 선택 가능한 모든 경우에 대하여 실행 중에 계산된다. Depth-first 트리 구조에서의 메모리 요구량은 전체 검색을 하기 위해 필요한 64×64=4096워드가 아니라 1088워드만 필요하다.

각각의 비트 rate 모드에서의 검색 구조는 유사하다. 다른 점은 펄스들의 수 그리고 거기에 따른 트리 검색의 레벨 수이다. 다른 코드북에 따른 비교 가능한 검색 복잡도를 유지하기 위해 테스트된 위치들의 수는 유사하게 유지된다.

Ⅲ. 개선된 depth-first 트리 검색 알고리즘

3.1 개선된 기준 신호 $b(n)$의 사용

AMR-WB에서는 비트율에 따라 스케일링 배율 α를 2~0.5 범위에서 일괄적으로 설정하였다. 이는 유성음과 무성음이 반복적으로 발생하는 음성의 특성을 고려하지 않고 비트율만을 고려하여 선택해 변화하는 유성음 무성음의 구분을 정확하게 결정하지 못한다. 이를 개선하고자 비트율에 따른 스케일링 배율 할당 방법을 버리고 해당 부프레임이 유성음인지 무성음인지를 결정하여 스케일링 배율을 부프레임 별로 변화 할 수 있도록 하였다. 이전 부프레임이 유성음인지 무성음인지를 조사하여 현재 프레임의 스케일링 배율을 선택하였다. 이는 유성음에서 무성음으로 무성음에서 유성음으로의 전이가 부프레임의 길이보다 충분히 길다는 것을 이용한 것으로 이전 프레임이 유성음 또는 무성음이라면 현재 프레임은 이전 프레임의 유성음 무성음 판단으로 사용하여도 무방하다.

식 (6)의 레퍼런스 벡터를 보면 장구간 예측 잔여신호인 r_{LTP}, 표적 신호 그리고 임펄스 응답의 상관도인 $d(n)$의 가중치 합으로 계산된 것을 알 수 있다. 두 번째 항 $d(n)$을 변경하여 사용하기도 한다[9]. 각 항의 가중치는 에너지의 비와 모드에 따라 2~0.5까지 선택하게 되어 있다. 레퍼런스 벡터를 구성하는 두 신호의 스펙트로그램을 보면 다음과 같다.

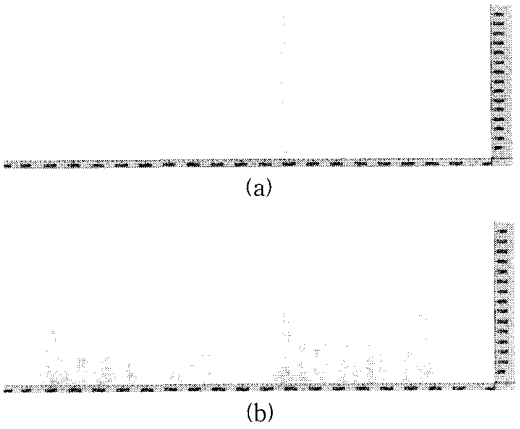


그림 4. 레퍼런스 벡터를 구성하는 신호의 스펙트로그램 (a) r_{LTP}의 스펙트로그램 (b) $d(n)$의 스펙트로그램
Fig. 4. Spectrogram of Reference Vector. (a) r_{LTP} (b) $d(n)$

<그림 4(a)>를 보면 r_{LTP}는 전 주파수 영역에 걸쳐 에너지가 분포되어 있음을 볼 수 있으며 이는 무성음 성분을 포함하고 있음을 알 수 있다. <그림 4(b)>에서 $d(n)$은 에너지가 저주파 영역에 집중되어 있으며 하모닉 성분을 포함하고 있다.

따라서 레퍼런스 벡터 $b(n)$을 계산할 때 해당 부프레임의 유성음 무성음인지를 고려하여 스케일링 배율을 설정한다. 제안하는 레퍼런스 벡터 $b'(n)$은 다음과 같다.

$$b'(n)=\sqrt{\frac{E_d}{E_r}}\, r_{LTP}(n)+\beta\, d(n) \qquad (7)$$

레퍼런스 벡터 $b(n)$의 형태는 동일하지만 스케일링 배율 α대신하여 새로운 스케일링 배율 β를 사용하였다. β는 유성음인 경우 0.5, 무성음인 경우 2.5의 값을 갖도록 하여 모드에 따라 결정되지 않고 이전 부프레임의 유무성음 여부에 따라 0.5 또는 2.5의 값을 가지도록 설계하였다. 이는 기존의 방법에 비해 모드에 따라 결정되었던 스케일링 배율을 음성 신호에 따라 스케일링 배율을 변화시킴으로써 레퍼런스 벡터를 표적 신호와 유사하게 하여 효율적으로 펄스의 위치를 찾게 하였다.

3.2 펄스위치 재검색 알고리즘

검색의 목적은 위에 기술한 방법을 통해 최적화된 N_p펄스 위치를 가지고 코드벡터를 결정하는 것이다. 제안하고자 하는 새로운 검색은 "pulse-position likelihood-estimate vector" b를 직접 사용한다. 레퍼런스 벡터 $b(n)$으로부터 초기 검색을 위한 펄스들의 위치를 얻어낸다. 각 트랙에 위치하여야할 펄스를 레퍼런스 벡터 $b(n)$에서 큰 값부터 선택하여 위치 검색을 위한 초기 펄스 배열을 만들어 낸다. 펄스 검색에 있어서 한 개 혹은 두 개의 펄스를 제외하고 나머지 검색되어 있거나 혹은 초기화된 펄스들의 위치를 가지고 새로운 펄스를 찾아낸다. 이 방법으로 원하는 만큼의 수준만큼 반복적으로 펄스의 검색이 가능하다.

위치 검색에 기반을 둔 반복 알고리즘은 한번에 가장 적당한 위치를 찾는 방법을 통해 모든 트랙 내에 있는 펄스의 위치를 얻어낸다. 이 알고리즘은 좀더 정확한 펄스의 위치를 찾아내기 위해 몇 번이고 반복해서 수행될 수 있다. 각각의 반복에서 각 펄스의 트랙 내에서의 가장 적절한 위치는 이전에 검색되거나 초기화되어 있는 다른 펄스들의 위치를 유지하면서 검색한다. 검색을 하기 위해 먼저 이전에 검색되었거나 레퍼런스 벡터 $b(n)$으로부터 초기화된 펄스들로부터 식 (4)의 검색 기준을 최대화하는 코드북을 계산한다.

여기 펄스 검색을 하기 위해 먼저 레퍼런스 벡터 $b(n)$으로 결정하여 사용한다. i번째 펄스의 사인을 레퍼런스 벡터의 해당 펄스로 사용을 한다. 레퍼런스 벡터는 식 (6)을 사용한다.

검색을 단순화하기 위해서 신호 $d(n)$과 행렬Φ는 미리 선택된 sign을 포함하기 위해 수정된다. $s_b(n)$은 $b(n)$의 sign이며 수정된 신호 $d'(n)$은 다음과 같다.

$$d'(n)=s_b(n)d(n),\quad n=0,\cdots,N-1 \qquad (8)$$

그리고 수정된 자기상관 매트릭스 Φ'는 다음과 같다.

$$\Phi'(i,j)=s_b(i)s_b(j)\Phi(i,j),\quad \begin{matrix} i=0,\cdots,N-1 \\ j=i,\cdots,N-1 \end{matrix} \qquad (9)$$

식 (8)와 식 (9)에 의해 Q_k의 분자의 R은 다음과 같이 수정된다.

$$R=\sum_{i=0}^{N_p-1} d'(i) \qquad (10)$$

그리고 분모의 E 는 다음과 같이 수정된다.

$$E=\sum_{i=0}^{N_p-1}\Phi'(m_i,m_i)+2\sum_{i=0}^{N_p-2}\sum_{j=i+1}^{N_p-1}\Phi'(m_i,m_j) \qquad (11)$$

검색의 목적은 최적화된 N_p 펄스 위치를 가지고 최적의 코드벡터를 결정하는 것이다. 식 (4)에서 언급했던 Q_k를 최대화하는 것을 기본적으로 선택하게 된다. 레퍼런스 벡터 $b(n)$으로부터 초기 펄스 위치를 결정하는 방법 이외에 미리 검색된 여기 펄스를 초기 펄스 위치로 대체할 수 있다.

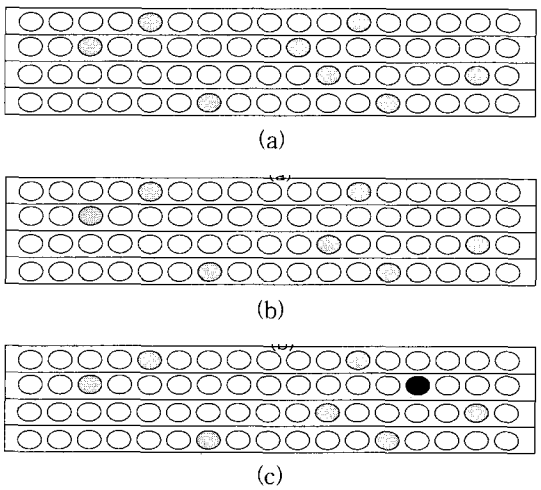


그림 5. 단일 펄스 검색 (a) 초기화된 펄스 (b) 하나의 펄스 제거 (c) 검색 된 펄스

Fig. 5. Single Pulse Search. (a) Init Pulses (b) Remove One Pulse (c) Add New Pulse

3.2.1 단일 펄스 재검색[10]

<그림 5>에서 보는 바와 같이 먼저 초기화되어 있는 여기 코드북의 펄스들로부터 중요도가 가장 낮은 펄스를 하나 제거한 후 새로운 펄스를 검색한다. 레퍼런스 벡터 $b(n)$으로부터 결정된 초기 펄스들의 위치로부터 Q_k를 계산한다. 새로운 펄스를 검색하기 위해 먼저 한 개의 펄스를 제거하고 남아 있는 펄스와 제거한 펄스가 속해 있는 트랙에서 새로운 펄스 하나를 추가하여 Q_k를 재계산한다.

최적화된 펄스를 결정하는 것이 검색의 목적이므로 새로 계산되어진 Q_k와 이전에 계산되어 있던 Q_k를 비교하여 큰 값을 선택을 하게 된다. 새로 추가된 펄스의 위치에 맞게 Q_k를 계산하고 다음 트랙에서 위의 과정을 반복적으로 수행한다. 계속적인 반복을 통해 모든 펄스의 위치를 재조정하며, 모든 펄스를 검색한 후에 새로 검색되어진 펄스들과의 상관관계를 고려하여 먼저 검색된 펄스들의 위치를 다시 검색할 수도 있다.

검색을 하기 위해 먼저 초기화되어 있는 펄스들로부터 Q_k를 계산한다. 식 (10)과 식 (11)에 의해 분자와 분모를 계산한다. 새로운 펄스를 검색하기 위해 먼저 한 개의 펄스위치를 제거한 뒤 제거된 후의 Q_k값을 계산해야 한다. 한 개의 펄스가 제거된 후의 검색 기준 $Q_k^{'}$ 의 분자와 분모는 다음과 같이 계산된다.

$$R^{'} = R - d^{'}(m) \tag{12}$$

$$E^{'} = E - \Phi^{'}(m,m) - 2R_{hv}(m) \tag{13}$$

다음 단계에서 분자와 분모는 새로 추가된 펄스의 변화량만큼 더해져 수정된다. 제거된 펄스가 속해 있는 트랙에서 새로운 펄스가 검색된다. 트랙의 펄스들에 대하여 새로운 Q_k 값을 최대화하는 펄스를 선택한다. R과 E는 다음과 같이 수정된다.

$$R = R^{'} + d^{'}(m) \tag{14}$$

$$E = E^{'} + \Phi^{'}(m,m) + 2R_{hv}(m) \tag{15}$$

$$v_h(n) = \sum_{i=0}^{k-1} h(n-m_i) \tag{16}$$

$$R_{hv}(m) = \sum_{n=m}^{N-1} h(n)v_h(n-m) \tag{17}$$

새로운 펄스를 검색하는데 있어서 $R_{hv}(m)$값은 해당 트랙에서 선택 가능한 모든 경우에 대하여 실행 중에 계산된다.

3.2.2 두 개의 펄스 재검색

두개의 펄스 검색은 3.2.1의 단일 펄스 재검색과 유사한 방법으로 수행된다. <그림 6>에서처럼 중요도가 낮은 두 개의 펄스를 제거하고 새로운 펄스를 검색하여 추가한다. 인접한 두 트랙에서 하나의 펄스씩 두 개의 펄스를 제거한 후 펄스가 제거된 트랙에서 새로운 펄스의 조합을 검색한다. 검색 기준은 단일 펄스검색의 경우와 동일하다. 검색을 하기 위해 먼저 이전에 검색되었거나 레퍼런스 벡터 $b(n)$으로부터 초기화된 펄스들로부터 식 (4)의 검색 기준을 최대화하는 코드북을 계산한다. 단일 펄스 재검색과 마찬가지로 검색을 하기 위해 먼저 레퍼런스 벡터 $b(n)$으로 결정하여 사용한다. 이미 설명했듯이 어떠한 방법으로든 초기화 되어있는 펄스위치들을 가지고 검색을 수행할 수 있다.

검색의 목적은 최적화된 N_p펄스 위치를 가지고 코드벡터를 결정하는 것이다. 위에서 언급했던 Q_k를 최대화하는 것을 기본적으로 선택을 하게 된다.

초기 펄스들의 위치로부터 Q_k를 계산한다. 새로운 펄스를 검색하기 위해 먼저 인접한 두 개의 트랙에서 각각 한 개씩 두 개의 펄스를 제거하고 남아 있는 펄스와 제거한 펄스들이 속해 있는 트랙에서 새로운 펄스를 각각 하나씩 추가하여 Q_k를 계산한다. 최적화된 펄스를 결정하는 것이 검색의 목적이므로 새로 계산되

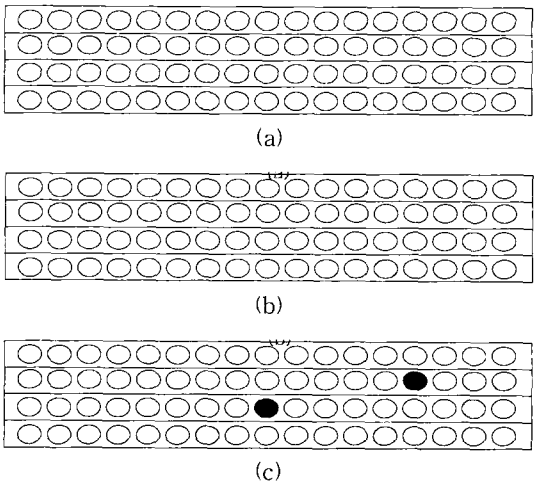


그림 6. 2 펄스 검색 (a) 초기화된 펄스 (b) 두개의 펄스 제거 (c) 검색 된 펄스

Fig. 6. Two Pulses Search (a) Init Pulses (b) Remove Tow Pulses (c) Add New Pulses

어진 Q_k와 이전에 계산되어 있던 Q_k를 비교하여 큰 값을 선택한다. 새로 추가된 두 펄스의 위치에 맞게 Q_k를 계산하고 이전 검색에 사용된 두 트랙 중 하나와 다른 인접한 트랙에서 위의 과정을 반복적으로 수행한다. 2 펄스 재검색 방법 역시 계속적인 반복을 통해 모든 펄스의 위치를 재조정하며, 모든 펄스를 검색한 후에 새로 검색된 펄스들과의 상관관계를 고려하여 먼저 검색된 펄스들의 위치를 다시 검색할 수 있다.

검색을 하기 위해 먼저 초기화되어 있는 펄스들로부터 Q_k를 계산한다. 식 (10)와 식 (11)에 의해 분자와 분모를 계산한다. 새로운 펄스를 검색하기 위해 먼저 인접한 트랙에서 각각 두 개의 펄스를 제거한 뒤 두 펄스가 제거된 후의 Q_k값을 계산한다. 두 펄스가 제거된 후의 검색 기준 $Q_k^{''}$의 분자와 분모는 다음 식 (18)과 식 (19)에 의해 다시 계산된다.

$$R^{''} = R - d^{'}(m_k) - d^{'}(m_{k+1}) \tag{18}$$

$$\begin{aligned} E^{''} = {} & E - \Phi^{'}(m_k, m_k) - \Phi^{'}(m_{k+1}, m_{k+1}) \\ & - 2\Phi^{'}(m_k, m_{k+1}) - 2R_{hv}(m_k) - 2R_{hv}(m_{k+1}) \end{aligned} \tag{19}$$

다음 단계에서 분자와 분모는 새로 추가된 펄스의 변화량만큼 더해져 수정된다. 제거된 펄스가 속해 있는 트랙에서 새로운 펄스가 검색된다. 트랙의 펄스들에 대하여 새로운 Q_k값을 최대화하는 펄스를 선택한다. R과 E는 다음과 같이 수정된다.

$$R = R^{''} + d^{'}(m_k) + d^{'}(m_{k+1}) \tag{20}$$

$$\begin{aligned} E = {} & E^{''} + \Phi^{'}(m_k, m_k) + \Phi^{'}(m_{k+1}, m_{k+1}) \\ & + 2\Phi^{'}(m_k, m_{k+1}) + 2R_{hv}(m_k) + 2R_{hv}(m_{k+1}) \end{aligned} \tag{21}$$

여기서 $R_{hv}(m)$은 식 (16)과 식 (17)로부터 구한다.

Ⅳ. AMR-WB 부호화기에 개선된 depth-first 트리 검색방법 적용

3장에 소개된 두 알고리즘을 사용하여 계산량을 줄이고 기존의 AMR-WB에서와 비슷한 성능을 갖는 고정 코드북 검색 방법을 제안한다. 먼저 레퍼런스 벡터를 계산하는데 있어 스케일링 배율을 3.1에서 제안한 유무성음에 따른 방법을 사용하였다. 해당 부프레임이 유성음인지 무성음인지를 결정하여 스케일링 배율을 프레임당 4번 변화 할 수 있도록 하였다. 이전 부프레임의 유무성음 여부에 따라 현재 프레임의 스케일링 배율을 선택하였다. 이는 유성음에서 무성음으로 무성음에서 유성음으로의 전이가 부프레임의 길이보다 충분히 길다는 것을 이용한 것으로 이전 프레임이 유성음 또는 무성음이라면 현재 프레임은 이전 프레임의 유성음 무성음 판단으로 사용하여도 무방하다.

AMR-WB에서 사용하는 depth-first 트리 알고리즘에 대체하여, 3.1에서 제안한 레퍼런스 벡터 $b(n)$을 사용하여 3.2에서 제안한 2펄스 재검색 후 단일 펄스 재검색을 수행한다. 각 모드 펄스의 개 수 만큼 개선된 레퍼런스 벡터의 최대값으로부터 트랙단위의 최대값을 갖는 펄스 배열을 추출함으로써 초기 펄스 배열을 생성한다. 반복 검색 알고리즘을 사용하기 위해서는, 펄스를 찾아가면서 추가되는 검색 기준이 주어지게 되는 AMR-WB의 depth-first 트리 알고리즘과는 달리, 초기화된 펄스들의 검색 기준 Q_k이 계산되어 있어야 한다.

레퍼런스 벡터 $b(n)$의 형태는 식(7)의 새로운 스케일링 배율β를 사용한다. 3.1에서 β는 유성음인 경우 0.5, 무성음인 경우 2.5의 값을 갖도록 고정하였으나 시뮬레이션을 통해 실험적으로 합성되는 음성의 음질을 최대로 하는 값을 사용하였다. 기존의 방법에 비해 모드에 따라 결정되었던 스케일링 배율을 음성 신호에 따라 스케일링 배율을 변화시켜 레퍼런스 벡터를 표적 신호와 유사하게 하여 효율적으로 펄스의 위치를 찾는다. 본 연구에서는 제안된 레퍼런스 벡터 $b(n)$으로부터 추출해낸 펄스 배열을 사용하여 2 펄스 재검색을 먼저 수행한 뒤 단일 펄스 재검색을 수행한다. 언급한 바와 같이 2 펄스 재검색과 단일 펄스 재검색 알고리즘은 좀더 정확한 펄스의 위치를 찾아내기 위해 몇 번이고 반복해서 수행될 수 있는 장점이 있다. 인접한 두 트랙에서 하나의 펄스씩 두 개의 펄스를 제거한 후 펄스가 제거된 트랙에서 새로운 펄스 조합을 검색한다. 이는 검색 기준 Q_k를 최대화하는 것을 선택함으로써 인접한 두 개의 트랙에서 기존에 검색된 펄스들과의 관계를 고려하여 가장 적절한 펄스를 검색하게 된다. 새로운 펄스를 검색하기 위해 먼저 인접한 두 개의 트랙에서 각각 한 개씩 두 개의 펄스를 제거하고 남아 있는 펄스와 제거한 펄스들이 속해 있는 트랙에서 새로운 펄스를 각각 하나씩 추가하여 Q_k를 계산한다. 트랙의 모든 펄스에 대해 Q_k값을 계산하지 않고 레퍼런스 벡터로부터 선정된 8개의 펄스만을 검색한다. 8×8번의 검

색을 수행하여 그중 가장 큰 Q_k값을 갖는 펄스를 검색한다. 새로 추가된 두 개의 펄스의 위치를 추가하여 Q_k를 수정하고 이전에 검색에 사용된 두 개의 트랙 중 하나와 다른 인접한 트랙에서 위의 과정을 반복적으로 수행한다. 모든 펄스를 검색한 후에 새로 검색된 펄스들과의 상관관계를 고려하여 먼저 검색된 펄스들의 위치를 다시 검색할 수 있다. 이에 대한 수식은 3.2.2 2펄스 재검색에 설명되어 있다.

위의 2 펄스 재검색을 통해 얻어진 펄스 배열을 가지고 단일 펄스 재검색을 수행한다. 각각의 반복에서 트랙내 각 펄스의 가장 적절한 위치는 트랙 내에서 이전에 검색되어 있는 다른 펄스들의 위치를 유지하면서 검색한다. 새로운 펄스를 검색하기 위해 먼저 한 개의 펄스 위치를 제거한 뒤 제거된 후의 Q_k값을 계산한다. 다음으로 새로 추가된 펄스의 변화량만큼 더해져 검색 기준 Q_k가 수정된다. 제거된 펄스가 속해 있는 트랙에서 새로운 펄스가 검색된다. 트랙의 펄스들에 대하여 새로운 Q_k값을 최대화하는 펄스를 선택한다. 이에 대한 수식은 3.2.1 단일 펄스 재검색에 설명되어 있다.

V. 시뮬레이션 결과

5.1 개선된 기준 신호 $b(n)$의 사용

3.1항에서 제안한 레퍼런스 벡터를 사용한 경우 다음과 같다. <표 1>은 3GPP와 ITU-T에서 제공하는 AMR-WB에서 사용하는 depth-first 트리 검색의 성능과 계산량을 보여주며 <표 2>는 제안된 레퍼런스 벡터를 사용하였을 때의 결과이다. 제시된 계산량은 deepth

표 1. AMR-WB depth-first 트리검색의 성능 및 계산량

Table 1. Performance and complexity of AMR-WB depth-first tree search algorithm.

모드	남자음성 (dB)		여자음성 (dB)		계산량
	SegSNR	SNR	SegSNR	SNR	
23.05	26.7677	25.8026	25.1301	23.5662	2×53×16=1696 loop
19.85	24.6081	23.5473	23.0145	21.1924	3×35×16=1680 loop
18.25	23.989	22.9437	22.2208	20.2111	3×36×16=1728 loop
15.85	22.016	21.2027	20.5169	18.9163	4×26×16=1664 loop
14.25	20.8269	20.0179	19.3461	17.9738	4×26×16=1664 loop
12.65	19.4996	18.9604	18.2773	17.1575	4×20×16=1280 loop
8.85	14.4435	15.9134	12.6729	14.2826	4×16×16=1024 loop

표 2. 스케일링 배율 β를 사용했을 경우의 성능 및 계산량

Table 2. Performance and complexity of using scaling factor β.

모드	남자		여자		계산량
	SegSNR	SNR	SegSNR	SNR	
23.05	27.024	25.9401	25.1401	23.602	2×53×16−1696 loop
19.85	24.9559	23.8989	23.1143	21.3779	3×35×16=1680 loop
18.25	24.113	22.9405	22.3893	20.7377	3×36×16−1728 loop
15.85	22.1715	21.3879	20.6204	18.9516	4×26×16−1664 loop
14.25	20.8677	20.2084	19.6085	17.8716	4×26×16−1664 loop
12.65	19.5885	19.0008	18.233	16.7947	4×20×16−1280 loop
8.85	14.4835	15.8872	12.688	14.1643	4×16×16−1024 loop

표 3. 단일 펄스 검색을 사용한 경우 성능과 계산량

Table 3. Performance and complexity of One pulse search.

모드	남자		여자		계산량
	SegSNR	SNR	SegSNR	SNR	
23.05	27.0796	25.6238	25.1446	23.3586	1×53×16+24×16−1232loop
19.85	24.7759	23.3993	22.904	21.017	1×35×16+18×16=843 loop
18.25	23.9542	22.7795	22.0948	20.2355	1×36×16+16×16−832 loop
15.85	21.857	20.9195	20.3355	18.9748	1×26×16+12×16=608 loop
14.25	20.5834	19.7523	19.1829	17.5897	1×26×16+10×16−576 loop
12.65	19.1971	18.7382	18.1348	16.957	1×20×16+8×16=448 loop
8.85	14.1396	15.7065	12.2895	14.2871	1×16×16+4×16−320 loop

표 4. 2 펄스 검색을 사용한 경우 성능과 계산량

Table 4. Performance and complexity of Two pulses search.

모드	남자		여자		계산량
	SegSNR	SNR	SegSNR	SNR	
23.05	27.093	25.493	25.2725	23.5582	1×53×16−8×12×16−2384 loop
19.85	24.7071	23.4711	23.1854	21.4201	1×35×16+8×9×16−1712 loop
18.25	23.8872	22.7101	22.3573	20.2374	1×36×16+8×8×16−1600 loop
15.85	21.9197	21.0516	20.4238	18.8157	1×26×16−8×6×16−1184 loop
14.25	20.6875	19.9609	19.2347	17.6539	1×26×16−8×5×16−1056 loop
12.65	19.3254	18.8978	18.2311	16.8236	1×20×16−8×4×16−832 loop
8.85	14.2539	15.8093	12.4023	13.9685	1×16×16+8×2×16−512 loop

first 트리 검색 방법에서 한번에 두 개의 펄스를 찾는 loop를 단위로 하였다.

제안된 레퍼런스 벡터를 사용하였을 경우 계산량의 감소는 없지만 기존의 AMR-WB보다 0.1~0.3dB 정도의 음질 향상을 보였다. 이는 레퍼런스 벡터를 유무성음에 따라 변화하므로써 무성음 구간에서 기존의AMR-WB 보다 원음에 가깝게 재현됨을 알 수 있다.

표 5. AMR-WB 스케일링배율 α와 제안된 스케일링배율 β

Table 5. Scaling factor of AMR-WB α and proposed β.

모드 (kbps)	AMR-WB의 스케일링 배율 α	제안된 스케일링배율 β	
		유성음일경우	무성음일경우
23.05	0.5	1.61	0.11
19.85	0.75	1.23	0.35
18.25	0.8	1.5	0.33
15.85	1.0	2.3	0.5
14.25	1.0	1.39	0.73
12.65	1.0	2.34	0.36
8.85	2.0	2.04	1.11

5.2 단일펄스 재검색과 2펄스 재검색

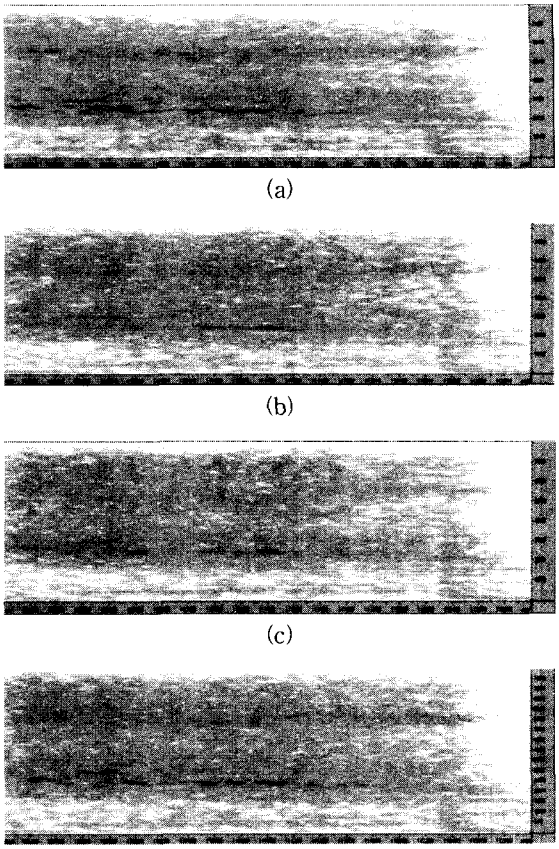
<표 3>과 <표 4>는 단일 펄스 재검색과 2펄스 재검색의 시뮬레이션 결과이다. AMR-WB의 depth-first 트리 검색을 1번 수행한 후 단일 펄스와 2 펄스 재검색을 사용할 경우 계산량이 AMR-WB와 유사한 경우 비슷한 성능을 보여준다.

위의 시뮬레이션 결과는 AMR-WB에서 사용하고 있는 depth-first 트리 검색 알고리즘을 사용한다. AMR-WB에서는 모드에 따라서 depth-first 트리 검색 방법을 1~4번 반복 사용하였는데 위의 시뮬레이션에서는 Detph-first 트리 검색 알고리즘을 1번 수행한 뒤 각각 단일 펄스 재검색과 2 펄스 재검색을 수행하였다. AMR-WB에서는 낮은 모드에서는 depth-first 트리 검색을 4번 수행 하는데 비해 제안된 방법은 한번 수행한 후 단일펄스와 2펄스 재검색을 수행하여 계산량의 많은 감소를 얻을수 있었지만 높은 모드에서는 depth-first 트리 검색을 수행하고 추가로 펄스 재검색을 수행하여 계산량의 증가는 피할 수 없다.

5.3 개선된 depth-first 트리 검색 알고리즘

최종적으로 AMR-WB의 depth-first 트리 검색 방법을 사용하지 않고 제안된 레퍼런스 벡터를 사용하여 바로 2펄스 재검색과 단일 펄스 재검색을 차례로 수행한 경우를 보면 <표 6>과 같다. 이는 AMR-WB에서 사용하고 있는 depth-first 트리 검색 방법을 제거한 후 4절에서 제안한 알고리즘을 사용하였다. 계산량은 AMR-WB에 비해 1/2~1/9가량 감소하였지만 음질 저하는 계산량의 감소에 비해 크지 않음을 볼수 있다. 모드에 따른 스케일링 배율 β는 <표 5>와 같다.

표에서 보는 바와 같이 AMR-WB와 제안된 방법은 성능에 있어서 큰 차이를 보이지 않지만 계산량에 있어서 1/2이상의 감소를 볼 수 있다.



(a) (b) (c) (d)

그림 7. 음성 스펙토그램 (a) 입력된 음성 (b) AMR-WB (c) 유무성음 구분 (d) 최종제안된 알고리즘

Fig. 7. Spectrogram of Voice. (a) Input Signal (b) AMR-WB (c) voice-unvoice (d) proposed algorithm

<그림 7>은 테스트에 사용된 음성중 무성음 부분의 스펙토그램이다. 점선으로 표시된 부분을 보면 최종 제안된 알고리즘이 AMR-WB의 결과 보다 원본음성의 에너지를 보다 충실히 재현됨을 보여주고 있다.

주관적 음질평가 MOS를 대신해 ITU-T의 P.862 PESQ(Perceptual Evaluation of Speech Quality)를 사용해 PESQMOS 값을 사용했다[11]. SNR의 결과와 비교해 볼 때 감소된 계산량에 비해 음질의 저하는 매우 적음을 볼 수 있다.

시뮬레이션에는 NTT에서 제공되는 16kHz 샘플링된 남성, 여성 각각 4명의 영어 음성을 사용하였다.

Ⅵ. 결 론

본 논문에서는, 유성음과 무성음에 따른 스케일링 값을 변화시키는 방법으로 레퍼런스 벡터를 추출하고, 이를 기반으로 위치 검색에 기반을 둔 반복 펄스위치 재검색 알고리즘으로 펄스의 위치를 재검색하는 고정 코드북 검색법을 설계하였다.

표 6. 최종적으로 제안된 방법의 성능과 계산량
Table 6. Performance and complexity of proposed Algorithm.

모드	남자		여자		계산량
	SegSNR	SNR	SegSNR	SNR	
23.05	26.8443	25.866	24.9242	23.4529	6×4×16+24×2×16=768
19.85	24.4589	23.8617	22.5981	21.347	4×4×16+18×2×16=576
18.25	23.8135	22.9757	21.9841	20.5416	4×4×16+16×2×16=512
15.85	21.8579	21.1333	20.0653	18.9151	3×4×16+12×2×16=384
14.25	20.4893	19.923	18.9257	17.8476	2×4×16+10×2×16=320
12.65	19.4376	19.0369	17.9916	16.9787	2×4×16+8×2×16=256
8.85	14.4419	15.9235	12.3801	14.2829	1×4×16+4×2×16=128

표 7. PESQ 음질평가
Table 7. PESQ test.

모드	AMR-WB		제안한 알고리즘	
	남자	여자	남자	여자
23.05	4.25675	3.97525	4.238	3.97175
19.85	4.23925	3.959	4.21175	3.89575
18.25	4.22075	3.9175	4.204	3.88025
15.85	4.14575	3.8405	4.14475	3.839
14.25	4.1255	3.82825	4.12725	3.7935
12.65	4.09175	3.7865	4.029	3.7375
8.85	3.80775	3.516	3.78975	3.50125

기존의 AMR-WB 음성부호화 방법에서는 여기 코드북 검색시 계산량을 줄이기 위해 depth-first 트리 검색 방법을 사용하며, 검색을 위한 레퍼런스 벡터는 피치 예측 잔여 신호와 표적 신호의 합으로 구성된 신호를 사용한다. 본 논문에서는 레퍼런스 벡터를 개선하고 코드북 검색 방법인 depth-first 트리 검색 방법을 개선하는데 중점을 두었다. 음성 프레임의 유무성음 종류에 레퍼런스 벡터를 달리 사용하는 방법을 제안하였는데, 레퍼런스 벡터를 구성하는 예측 잔여신호와 표적 신호의 가중치 합을 수행할 때 스케일링 계수를 유무성음에 따라 달리 하는 방법으로 계산량의 증가 없이 약 1 dB의 SEGSNR 개선을 이룰 수 있었다. 최적의 값을 찾기 위하여 다양한 레퍼런스 벡터를 사용하여 여기 코드북 검색을 수행하였다. 또한, 여기 코드북 검색시 단지 하나의 트리 구조를 검색하는 depth-first 트리 검색 방법을 개선하여 레퍼런스 벡터를 통해 후보 벡터를 결정하고 이를 이용하여 한 개 혹은 두 개의 펄스를 제거하고 남은 펄스들의 영향을 고려하여 새로운 펄스를 찾아 벡터를 수정하는 펄스 위치 재 검색 방법을 통해 최적의 여기 코드북을 검색하는 방법을 제안하였다. 제안된 방법은 depth-first 트리 검색 방법에 비해 반 이상의 적은 계산량에서 비슷한 성능을 얻었다.

참 고 문 헌

[1] ITU Radiocommunication Assembly, Document 8/1045-E, "Detailed Specifications of the Radio Interfaces of IMT-2000," Document 8/126 (Rev.1) April. 2000.

[2] Boudreau, D., Caire, G., Corazza, G.E., De Gaudenzi, R., Gallinaro, G., Luglio, M., Lyons, R., Romero-Garcia, J., Vernucci, A., Widmer, H. "Wide-band CDMA for the UMTS/IMT-2000 satellite component", Vehicular Technology, IEEE Transactions on, vol. 51 pp. 306~331, March 2002.

[3] ISO/JTC1/CS29/WG1 N2203, "Committee Draft International standard of ISO/IEC 14496-3", 1998.

[4] ISO/IEC JTC1/SC29/WG11 no.71, "coding of Moving Pictures and Associated Audio for Digital Stroage Media at up to about 1.5 Mbit/s DC11172-3 (Part 3. MPEG Audio)"

[5] ETSI draft EN 301 704, "Digital cellular telecommunication system; Adaptive Multi-Rate(AMR) speech transcoding"

[6] ITU-T Recommendation G.722.2, "Wideband coding of speech at around 16kbit/s using adaptive multi-rate wideband(AMR-WB)", January 2002.

[7] 3GPP TS 26.171, "AMR Wideband Speech Codec ; General description".

[8] 3GPP TS 26.173, "AMR Wideband Speech Codec ; ANSI-C code".
[9] Nam Kyu Ha, "A fast search method of algebraic codebook by reordering search sequence", Proc. in ICASSP, 1999.
[10] Hochong Park, "Efficient codebook search method ofr EVRC speech codec," IEEE Signal Processing Letters, vol.7, no.1, 2000.
[11] ITU-T P.862 (2001), "Perceptual evaluation of speech quality (PESQ) : An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs"

저 자 소 개

許 熺(學生會員)

2001년 2월 : 충북대학교 정보통신공학과 학사. 2001년 3월~현재 : 충북대학교 전파공학과 석사과정 <주관심분야 : 디지털신호처리, 음성신호처리>

李 寅 誠(正會員)

1983년 2월 : 연세대학교 전자공학과 학사. 1985년 2월 : 연세대학교 전자공학과 석사. 1992년 12월 : Texas A&M University 전기공학과 박사. 1986년 5월~1987년 7월 : 한국통신 연구개발단 전임연구원. 1993년 2월~1995년 9월 : 한국전자통신연구원 이동통신기술연구단 선임연구원. 1995년~현재 : 충북대 전기전자공학부 부교수. <주관심분야 : 음성 및 영상신호압축, 이동통신, 적응필터>

池 德 求(正會員)

1998년 2월 : 충북대학교 전자공학과 학사. 2000년 2월 : 충북대학교 전파공학과 석사. 1999년 12월~8월 : ㈜ DSP D&C 연구원. 2000년 8월~현재 : 한국전자통신연구원 이동A/V 연구팀 연구원. <주관심분야 : 이동통신, 음성신호처리>

尹 炳 植(正會員)

1990년 2월 : 경북대학교 전자공학과 학사. 1992년 2월 : 경북대학원 전자공학과 석사. 1992년 1월~현재 : 한국전자통신연구원 이동A/V 연구팀 선임연구원. <주관심분야 : 이동통신, 음성신호처리, 영상신호처리>

崔 松 仁(正會員)

1982년 2월 : 광운대학교 응용전자공학과 학사. 1987년 2월 : 광운대학원 전자계산기공학과 석사. 1982 7월~현재 : 한국전자통신연구원 이동A/V 연구팀 책임연구원. <주관심분야 : 이동통신, 음성신호처리, 영상신호처리>