Block Filtering과 QFT를 이용한 실시간 음장 효과구현

손성용(ICU), 서정일(ETRI), 한민수(ICU)

<차 례>

<Abstract>

Real-Time Sound Field Effect Implementation Using Block Filtering and QFT

Sung-Yong Sohn, Jeongil Seo, Minsoo Hahn

It is almost impossible to generate the sound field effect in real time with the time-domain linear convolution because of its large multiplication operation requirement. To solve this, three methods are introduced to reduce the number of multiplication operations in this paper. Firstly, the time-domain linear convolution is replaced with the frequency-domain circular convolution. In other words, the linear convolution result can be derived from that of the circular convolution. This technique reduces the number of multiplication operations remarkably. Secondly, a subframe concept is introduced, i.e., one original frame is divided into several subframes. Then the FFT is executed for each subframe and, as a result, the number of multiplication operations can be reduced. Finally, the QFT is used in stead of the FFT. By combining all the above three methods into our final the SFE generation algorithm, the number of computations are reduced sufficiently and the real-time SFE generation becomes possible with a general PC.

* Keywords : Sound field effect, convolution, block filtering, Quick Fourier Transform

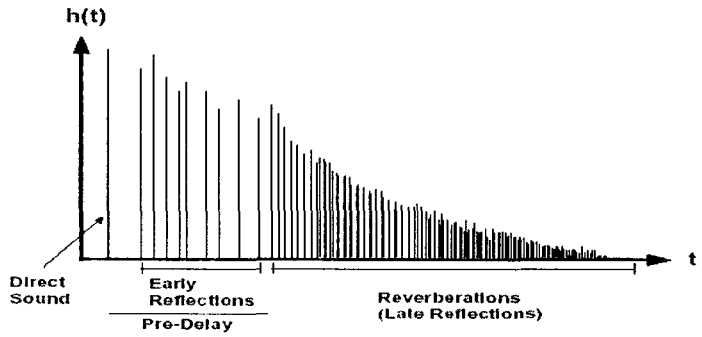
1.서 론

최근 10여년 동안 사운드 제어와 음장효과 등의 분야에서 많은 연구원들의 연구가 있어왔다[1]. 이러한 연구의 가장 큰 동기는 청취자들의 음에 대한 보다 사실적이고 양질을 갖는 음의 요구이다. 예를 들어 청취자의 취향과 기분에 맞추어 고주파 또는 저주파대역을 강조하길 원하고, 혹은 자신은 거실에 앉아 있지만 콘서트홀에서 듣는 것처럼 느끼길 원한다. 이 논문에서는 후자인 원 음에 음장감을 부여하여 자신이 마치 다른 장소에서 음악을 듣는 기분을 느끼게 하는 음장효과에 대하여 다룰 것이다. 음장효과는 시간 영역에서 원 신호와 룸 임펄스 신호사이의 선형 복적분을 이용하여 쉽게 구현 될 수 있다. 그러나 선형 복적분을 이용한 음장효과의 구현은 엄청난 곱의 연산량을 필요로 하기 때문에 실시간 구현이 거의 불가능하다. 이러한 단점을 해결하기 위해 기존 연구에서는 룸 임펄스를 임의적으로 수정함으로써 곱의 연산량을 줄여왔다. 그러나 공간의 특성을 가지고 있는 룸 임펄스의 신호를 임의적으로 수정함으로써 공간특성을 상당 부분 상실하게 되어 풍부한 음장감을 부여할 수 없으며 또한 양질의 음장효과를 가져오기 힘들다. 또 하나의 방법으로 주파수 영역에서 순환 복적분을 수행하여 선형 복적분의 결과를 가져오는 것이다. 이를 수행하기 위해서는 원 신호와 룸 임펄스 신호를 한번에 주파수 영역으로 전환을 해야 한다. 그러나 실제 현장에서 녹음되는 룸 임펄스의 데이터 수는 샘플링 주파수가 44.1 kHz이고 실행 시간이 2초라 가정 할 경우 최소한 88,100개 이상의 데이터를 한 번에 주파수 영역으로 전환해야 한다. 이것은 현재 PC환경에서는 어려운 일이다. 본 논문에서는 기존의 연구 방법의 단점을 해결하여 모든 룸 임펄스의 정보를 이용하여 실시간으로 음장효과를 구현할 것이다. 이는 시간 영역에서의 선형 복적분의 결과를 주파수 영역의 순환 복적분으로부터 얻어냄으로써 상당량의 곱의 연산량을 줄일 수 있으며 또한 하나의 프레임을 여러 개의 서브 프레임으로 나눔으로써 데이터의 길이에 상관없이 주파수 영역에서 음장효과 구현을 가능케 하며 또한 연산량 감소의 효과를 가져 올 것이다. 마지막으로 주파수 영역으로의 전환을 위해 FFT대신 QFT를 이용함으로써 연산량을 줄일 수 있다.

본 논문의 구성은 2장에서 음장효과의 정의와 구현 방법과 필요한 연산량에 대해 알아볼 것이며 3장에서는 제안된 방법의 간단한 정의와 곱의 연산량을 계산할 것이다. 4장에서는 제안된 방법을 이용하여 실제 음장효과 구현 방법에 대해 알아볼 것이며 5장에서는 실험 및 결과, 마지막 6장에서 결론을 맺을 것이다.

2. 음장효과의 정의 및 구현

음장효과는 무향실에서 녹음된 원음에 콘서트홀 혹은 동굴에서 발생하는 울림을 부여함으로써 원음에 음장감을 만드는 것이다. 음장효과에 부여할 울림 음은 울림이 존재하는 공간에서 임의의 임펄스성 신호를 발생시켜 얻을 수 있다. 이러한 울림 음을 룸 임펄스라 한다[2].



<그림 1> 룸 임펄스의 구성

<그림 1>에서 보는 것과 같이 룸 임펄스는 크게 두 부분으로 모델링이 가능하다. 룸 임펄스의 시작부분을 초기 반사(Early reflection), 크기가 상당히 떨어지는 부분을 추후 반사(Reverberation)라고 한다. 초기 반사는 음원에서 청취자 간에 존재하는 반사체에 의한 직접 반사를 나타내고 있으며, 추후 반사는 그러한 반사파가 다시 반사의 반사를 거듭한 끝에 난반사의 형태를 띠어 청취자에게 도달하는 것을 의미한다. 음장효과의 구현은 시간 영역에서 원 신호와 룸 임펄스 신호사이의 선형 복적분을 이용하여 쉽게 구현 될 수 있다. 선형 복적분을 이용하여 음장효과를 구현 할 경우, 원 신호 $x(n)$의 길이를 N_1이라 하고 룸 임펄스 신호 $h(n)$의 길이를 N_2라 할 경우 식 (1)에서 보는 것과 같이 $h(-m)$의 신호를 순차적으로 이동하여 각각의 데이터의 곱과 그 곱의 합의 연산으로써 이루어진다.

$$y(n) = \sum_{m=-\infty}^{\infty} h(n-m)x(m) \tag{1}$$

따라서 선형 복적분의 곱의 연산량은 각각의 $x(n)$ 신호마다 모든 $h(n)$의 데이터가 곱해지기 때문에 곱의 연산량은 $N_1 \times N_2$이 된다. 곱의 연산량 식에서 알 수 있듯이 만약 N_1의 길이가 1이 증가 한다면 곱의 연산량은 N_2 만큼 증가하게 된다.

이러한 특징으로 인해 N_1, N_2의 길이가 길어질수록 곱의 연산량은 급속도로 증가하게 되어 실시간 구현을 불가능하게 만드는 원인이 된다.

3. 제안된 음장효과의 실시간 구현 방법

3.1 주파수 영역에서의 순환 복적분

시간 영역에서 선형 복적분의 결과는 주파수 영역에서 순환 복적분을 통해 얻어질 수 있다. 우리는 이미 DFT (Discrete Fourier Transform)를 이용하여 시간 영역에서의 복적분을 주파수 영역에서의 곱셈으로 전환할 수 있다는 사실을 알고 있다. 이러한 특성을 이용하여 복적분에 필요한 곱의 연산량을 줄일 수 있다. 순환 복적분으로부터 선형 복적분의 결과를 얻는 방법에는 두 가지가 있다. 첫 번째 방법은 오버랩에드(Overlap-add) 방법이다. 이 방법은 다음 절에서 언급할 블록 필터링과 DFT를 거친 블록들의 결과를 서로 더함으로써 수행이 된다. 또 하나의 방법은 오버랩세이브(Overlap-save) 방법이다. 오버랩세이브 방법은 오버랩에드에서 더하는 연산 대신 세이브를 하는 것으로 수행을 할 수가 있다. 여기서 우리는 오버랩세이브 방법을 택할 것이다. 오버랩세이브를 수행하기 위해서 각각의 신호 $x(n)$과 $h(n)$은 FFT 변환크기로 분할 또는 확장되어야 한다. 먼저, 원 신호 $x(n)$을 백터열 $\mathbf{X} = \{x(0), x(1), \ldots, x(N_1 - 1)\}^T$ 과 $h(n)$을 응답길이 N_2인 백터열 $\mathbf{h} = \{h(0), h(1), \ldots, h(N_2 - 1)\}^T$로 나타낼 수 있다. 여기서 T는 전치 행열 변환 연산자이다. 이때 $N_1 \geq N_2$과 $N_1 = 2^r (r = 1,2,...)$을 만족하는 임의의 N_1에 대해서 $\mathbf{X}$는 $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_L\}$와 같이 서브 프래임 벡터열로 표현될 수 있으며, $\mathbf{x}_k$는 다음과 같이 나타낸다.

$$\mathbf{x}_k = \{0,...,0, x(N_1 - N_2 + 1),..., x(N_1 - 1)\}^T, k = 0 \quad (2)$$

$$\mathbf{x}_k = \{x(k(N_1 - N_2 + 1)),..., x(N_1 + k(N_1 - N_2 + 1)\}^T \quad k \neq 0 \quad (k = N_1 / L - 1) \quad (3)$$

이때 $N_1 > N_2$일 경우 벡터열 $\mathbf{h}$는 다음과 같이 N_1의 크기의 백터열 $\hat{\mathbf{h}}$로 확장되어야 한다.

$$\mathbf{h} \cong \hat{\mathbf{h}} = \{h(0), h(1), \ldots, h(N_2 - 1), 0, \ldots, 0\}^{\mathrm{T}} \tag{4}$$

오버랩세이브 방법은 $\hat{\mathbf{h}}$과 $\mathbf{x}_k$의 주파수 영역연산으로부터 출력 신호열 $\mathbf{y}_k$를 구함으로써 선형 복적분을 구현할 수 있다. 이는 다음과 같이 나타낸다.

$$y_k = \mathrm{IFFT}\left\{\mathrm{FFT}\{\mathbf{x}_k\} \circ \mathrm{FFT}\{\hat{\mathbf{h}}\}\right\} \tag{5}$$

여기서 연산자 ∘는 각각의 벡터열의 곱을 나타낸다. 이때 FFT 수행 후의 각각의 벡터열의 곱은 시간영역에서 두 벡터열의 순환 복적분으로 나타난다[3]. $\hat{\mathbf{h}}$과 $\mathbf{x}_k$의 시간영역에서 순환 복적분을 고려하면, $\mathbf{y}_k$를 다음과 같이 시간영역에서 처리하여야 한다.

$$\mathbf{y}_k = \{0, \ldots, 0, y(N_1 - N_2 + 1), \ldots, y(N_1 - 1)\}^{\mathrm{T}} \tag{6}$$

이로써 순환 복적분은 선형 복적분연산으로 대치될 수 있다. 이때 출력 신호열 $y(n)$는 벡터열 $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_L\}^{\mathrm{T}} = \{y(0), y(1), \ldots, y(N_1 - 1)\}^{\mathrm{T}}$로 얻어진다. 오버랩세이브의 곱의 연산량을 계산해 보면 다음과 같다. 오버랩 되는 길이 $M-1$을 N_2의 길이와 동일하게 정의 하면 필요한 곱의 연산량은 식 (7)과 같다.

$$(2N_1 + N_2) \times \log_2(2N_2) + 8N_1 / N_2 \tag{7}$$

여기서 $N_2 \le N_1$이며 FFT의 크기는 $2N_2$가 된다. 식 (6)의 결과는 우선 2N-Point FFT에 필요한 곱의 연산량은 $N \log_2(2N)$로 정의 된다[3]. 오버랩세이브를 수행하기 위해서는 원 신호와 롬 임펄스의 FFT와 두 신호의 곱의 결과의 IFFT로 총 3번의 FFT 를 수행하게 된다. 그러므로 FFT 를 위한 곱의 연산량은 $(N_2 \times \log_2(2N_2) \times (2N_1 / N_2 + 1)$이 된다. 이렇게 주파수 영역으로 전환된 데이터들은 복적분 대신 곱의 연산으로 바뀌게 되므로 $2N_1 \times (N_2 / N_1)$만큼의 곱의 연산량이 필요하게 된다. 마지막으로 주파수 영역에서의 곱은 (a+bj)(c+dj)으로 연산 되기 때문에 ac, ad, bc와 bd 이렇게 4번의 곱이 수행 되어 결과적으로 주파수 영역에서의 총 곱의 연산량은 4를 곱해준 식 (7)로 정의 될 수 있다. <표 1>에서 시간 영역에서선형 복적분의 곱의 연산량과 주파수 영역에서의 곱의 연산량을 비교하였다.

<표 1> 영역변화에 따른 곱의 연산량

N_2 ($N_1 = 2N_2$)	시간 영역	주파수 영역	비율
8	64	176	0.38
128	16,384	5,136	3.19
512	262,144	25,616	10.23
2048	4,194,304	122,896	34.14

<표 1>에서 볼 수 있듯이 N_2의 길이가8인 경우 오히려 연산량이 증가 하였다. 그러나 N_2의 길이가 커질수록 곱의 연산이 줄어든 것을 알 수가 있다. 그러므로 N_2의 길이가 충분히 크다면 오버랩세이브 방법을 이용하여 곱의 연산에서 상당한 이득을 볼 수 있다. <표 1>에서의 비율은 주파수 영역에서의 연산량으로 시간 영역에서 얻어진 연산량을 나누어 줌으로써 구했다.

3.2 블록 필터링

음장효과 구현을 위한 복적분은 블록 필터링을 이용하여 구현이 가능하다. 블록 필터링을 이용하는 이유는 서론에서 언급한 바와 같이 한번에 처리해야할 데이터의 양이 많을 때 이를 나누어 수행함으로써 데이터의 길이에 대한 제약을 없애기 위함이다. 원 신호와 룸 임펄스 신호를 영을 포함하지 않는 길이 L을 갖은 블록으로 나눈다고 가정한다. $x(n)$은 식 (8)에서 보는 것과 같이 k개의 $x_k(n)$으로 나타낼 수 있다.

$$x_k(n) = x(n) \quad kL \le n \le (k+1)L-1$$
$$0 \qquad O/W \tag{8}$$

$$x\ (n) = \sum_{k=0}^{\infty} x_k(n) \tag{9}$$

$x(n)$은 식 (9)로 다시 나타낼 수 있으며 식 (9)를 이용하여 복적분을 다음과 같이 나타낼 수 있다.

<표 2> 블록 필터링 적용 시 곱의 연산량

N_2 ($N_1=2N_2$)	L	주파수 영역	블록 필터링	감소율
8	4	176	96	1.83
128	64	5,136	1,296	3.96
512	256	25,616	5,136	4.98
2,048	1,024	122,896	20,496	5.99
4,096	2,048	266,256	40,976	6.49

$$x(n)\otimes h(n)=\sum_{k=0}^{r} x_k(n)\otimes h(n) \tag{10}$$

블록 필터링을 이용하여 복적분을 수행할 때 곱의 연산량을 계산하면 식 (11)과 같다.

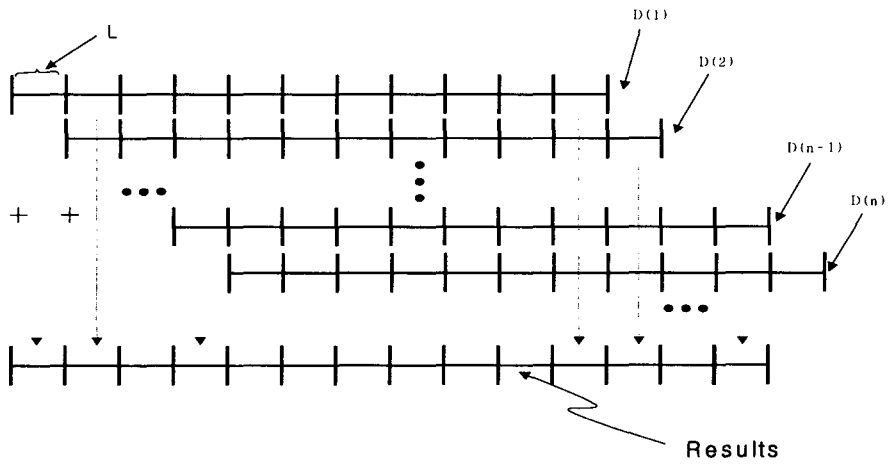
$$(2N_1+N_2)\times\log_2(2N_2/L)+8N_1/N_2 \tag{11}$$

식 (11)은 원 신호가 L크기를 갖는 여러 개의 블록으로 나누어져 FFT를 수행할 경우 곱의 연산량은 $N/L\log_2(N/L)\times L$ 이 되므로 이를 정리하여 식 (7)에 적용하여 얻어 진다. 실제 곱의 연산량을 계산하면 <표 2>와 같다. N_1이 커질수록 곱의 연산량에 대한 이득이 커짐을 알 수가 있다. 여기서 블록의 크기 L를 N_1의 절반 값으로 택한 이유는 <표 3>에서 보는 바와 같이 L의 크기가 작을수록 곱의 연산량이 증가 하는 것을 알 수 있다. 이는 L의 크기가 작을수록 지연되어 나오는 결과의 시퀀스가 많이 지게 되어 곱의 연산량이 많아지기 때문이다. 예를 들어 원 신호와 룸 임펄스의 신호의 길이가 20이고 블록의 크기를 10으로 가정 했을 경우 지연된 시퀀스는 2개가 된다. 즉 <그림 2>에서 $D(n)$이 $D(1), D(2)$이 존재하게 된다. 그러므로 곱해져야 할 블록의 수가 10개짜리 한 개가 되므로 신호간의 곱의 연산량은 10이 된다. 반면에 블록의 크기를 5로 하였을 경우 지연된 퀀스는 4개가 된다. 즉 <그림 2> 에서 $D(n)$이 $D(1),\ldots,D(4)$까지 존재하게 된다. 이 경우 곱해야 할 블록의 수가 5개짜리 9개가 되므로 필요한 곱의 연산량은 45개가 된다. 결과적으

로 블록의 크기가 5개일 때가 10개일 때 보다 4.5배나 많은 곱의 연산량을 필요로 하게 된다.

<표 3> 블록 크기에 따른 곱의 연산량

N_2	L	주파수 영역+ 블록 필터링
4,096	4	225,296
4,096	64	143,376
4,096	256	102,416
4,096	1,024	61,456
4,096	2,048	40,976



<그림 2> L크기에 따른 지연 시퀀스

3.3 QFT

곱의 연산량을 줄이기 위해 시간 영역의 복적분을 주파수 영역의 곱의 연산으로 전환하기 위해 FFT를 사용하였다. 여기서 FFT 대신 QFT (Quick Fourier Transform)을 이용하여 곱의 연산량을 줄 일수 있다[4]. 주파수 영역으로의 전환 알고리즘은 <표 4> 에서 보는 것과 같이 많은 종류가 있으며 이들 각각의 알고리즘들은 서로 다른 장단점을 가지고 있다. 예를 들어 RAD-2 알고리즘(Radix-2알고리즘: the Cooley-Tukey algorithm)은 메모리 사용량 측면에서 다른 알고리즘보다 적어 메모리에 중점을 둔 분야에 적합한 반면 곱의 연산량은 상대적으로 많아 연산량에 중점을 둔 분야에서는 부적합하다. 또한 QFT의 경우 메모리 사용량이 상대적

<표 4> 1024-FFT 를 위한 연산량과 메모리사용

Algorithm	Memory usage (byte)	The number of computation(times)		
		Float Mults	Integer Mults	Binary Shifts
RAD2	72440	20480	2048	1023
RAD4	72536	14336	3071	277
SRFFT	72508	5522	2542	1988
FHT	72652	8841	2048	12
QFT	122072	2560	1048	144
DIFT	78632	17664	1076	1074

으로 많은 단점을 가지고 있지만 곱의 연산량이 적어 음장효과 구현에 적합하다. <표 4>에서 RAD-4 알고리즘은 Radix-4의 약자이며, FHT는 Fast Hartley Transform, SRFFT 는 Split Radix FFT, FHT는 Fast Hartley Transform이며 DIFT는 Decimation in Time Frequency FT의 약자이다. QFT은 다음과 같이 간략히 정의할 수 있다. DFT 는 식 (12)와 같이 정의할 수 있다. 식 (12)의 $e^{-j2\pi nk/N}$ 은 식 (13)으로 나타낼 수 있으며 $x(n)$은 식 (14)로 다시 나타낼 수 있다. 그러므로 식 (12)는 식 (15)와 같이 나타낼 수 있다.

$$c(k)=\sum_{n=0}^{N-1}x(n)e^{-j2\pi nk/N} \qquad k=0,...,N-1 \tag{12}$$

$$e^{-j2\pi nk/N}=\cos(2\pi nk/N)-j\sin(2\pi nk/N) \tag{13}$$

$$x(n)=r(n)+jv(n)=[u_e(n)+u_o(n)]+j[v_e(n)+v_o(n)]) \tag{14}$$

$$c(k)=\sum_{n=0}^{N-1}(u+jv)(\cos-j\sin) \tag{15}$$

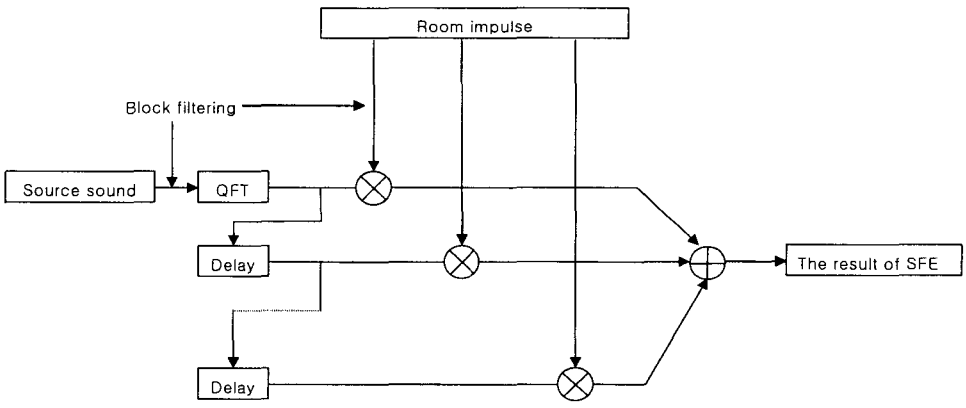
$$c(k)=\sum_{n=0}^{N/2-1}[u_e\cos+v_o\sin]+j[v_e\cos-u_o\sin] \tag{16}$$

식 (15)에서 기함수의 합은 영이 되므로 식 (16)과 같다. 그러므로 곱의 연산량이 반으로 줄게 된다.

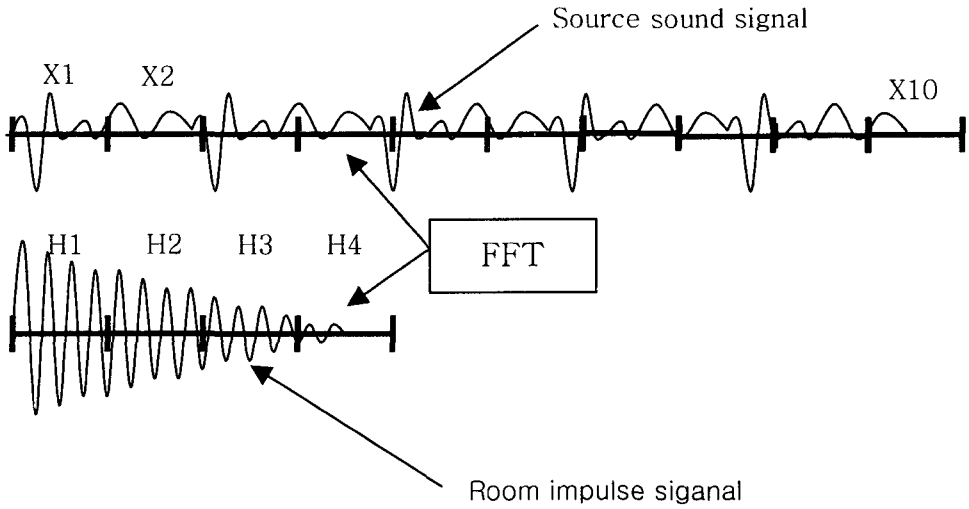
4. 실시간 음장효과 구현

제안된 방법을 이용하여 음장 효과를 구현하기 위한 흐름도는 <그림 3>과 같다. 원 신호와 룸 임펄스 신호가 들어 오면 <그림 4>와 같이 블록 필터링을 거친다. 첫번째 블록은 <그림 5>과 같이 블록의 크기만큼 0으로 이루어진 블록을 앞부분에 더해 준다. 같은 방법으로 마지막 블록까지 수행 한다. 그 다음 블록연산에서는 0으로 이루어진 블록이 있던 자리에 $x1$을 넣고 이전의 $x1$자리에는 $x2$를 넣는다. 다음으로 룸 임펄스의 경우는 블록의 크기만큼 뒤에 0으로 이루어진 블록을 더해준다. 블록 필터링을 거친 데이터들은 <그림 3>에서 보는 것과 같이 QFT를 이용하여 주파수 영역으로의 전환이 이루어진다. 주파수 영역에서 각각의 블록은 곱의 연산을 수행함으로써 시간 영역의 복적분 연산을 수행하게 된다.

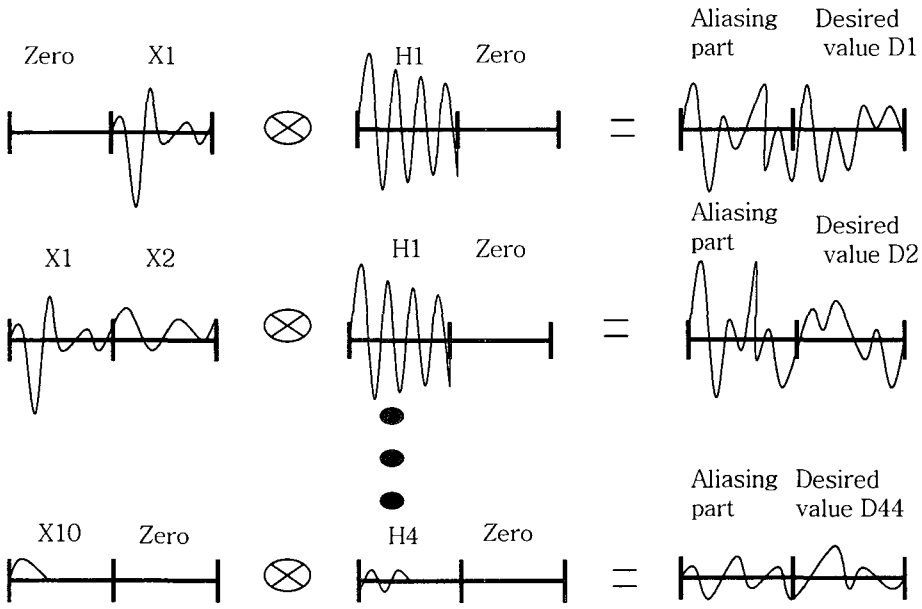
곱의 연산이 수행된 각각의 블록들은 IQFT를 통해 시간영역으로의 전환이 이루어진다. IQFT를 거친 결과는 <그림 5>의 오른쪽 블록 그림과 같다. 블록의 처음 부분은 엘리어싱 부분으로 의미가 없는 값이며, 뒷 부분의 결과가 우리가 얻고자 하는 결과이다. 이렇게 만들어진 시퀀스들은 덧셈 연산을 수행함으로써 최종 결과가 나오게 된다.



<그림 3> 음장효과 구현 흐름



<그림 4> 신호의 블록



<그림 5> Zero-padding을 이용한 오버랩세이브 방법

5. 실험 및 결과

5.1 실험 환경

실험 환경은 <표 5>와 같다. 원 신호는 무향실에서 트럼펫연주를 녹음하였으며 룸 임펄스는 장충 체육관에서 임펄스성 신호를 발생하여 녹음하였다. 신호의 샘플링 주파수는 44.1 kHz이며 모노 신호이다. 원신호의 실행 시간은 18.5초 이며 데이터 수는 819,200개이다. 룸 임펄스의 실행 시간은 2.9초이며 데이터 수는 129,687개이다. 실험을 위한 컴퓨터의 CPU는 500 MHz이다.

<표 5> 실험 환경

	원 신호	룸임펄스
녹음 장소	무향실	장충 체육관
샘플링 주파수	44.1 kHz	44.1 kHz
모드	모노	모노
실행 시간	18.5 초	2.9 초
데이터 량	819,200 개	129,687 개
CPU	500 MHz	

5.2 실험 결과

실험은 총 3회에 걸쳐 수행한 실행 시간의 평균값을 이용하여 수행하였다. 실험 결과 시간영역에서 복적분을 이용하여 음장효과를 실행하였을 때 걸린 시간은 약 80분이다. 반면에 오버랩세이브와 신호의 서브 프레임화를 거쳐 음장효과를 실행하였을 때 걸린 시간은 17초이다. 이는 곱의 연산량의 감소로 인해 79분 43초, 약 282배의 시간을 단축할 수 있었다. <표 6>은 FFT 수행 시간과 QFT 수행 시간을 제외한 데이터들의 곱과 합의 수행 시간을 QFT 크기에 따라 나타내었다. 여기서 총 수행 시간을 결정하는데 비중을 차지하는 요소를 분석해 보면, QFT 크기가 작을수록 QFT를 수행하기 위한 실행 시간의 비중은 거의 없는 반면 순수 연산 시간의 비중이 커지게 된다.

· 이는 4장에서 언급한 바와 같이 QFT 크기가 작을수록 IQFT 이후, 곱 연산의 급증으로 수행 시간이 길어지기 때문에 QFT 크기가 작을수록 순수 연산의 수행 시간이 커지게 된다. 또한 QFT 크기가 커질수록 QFT를 수행하기 위한 시간은 증가하여 비중이 커지는 반면 순수 연산 시간은 줄어 총 수행 시간을 결정하는데 비중이 작아진다. 이러한 결과 QFT 크기가 어느 이상 커지게 되면 총 수행 시간

이 다시 증가하게 된다. 그러므로 적절한 QFT 크기의 선택이 요구 된다.

<표 6> 실험 결과

		QFT 제외 한 수행시간	*QFT* 수행 시간	총 수행 시간
선형복적분	.	4802 초	.	4802 초
주파수 영역 & 프레임화 (QFT 크기)	512	121 초	0.81 초	121 초
	2,048	29 초	6 초	35 초
	8,192	**10 초**	**7 초**	**17 초**
	32,768	4 초	10 초	14 초
	65,536	2 초	16 초	18 초

6. 결 론

본 논문의 결과 시간영역에서 선형 복적분을 수행함으로 해서 발생하는 연산량 문제를 세 가지 방법을 이용하여 상당 부분 해결하였다. 첫 번째 방법은 시간영역에서의 선형 복적분을 주파수 영역의 오버랩세이브 방법을 이용함으로써 연산량 감소를 유도하였다. 여기서 두 데이터의 길이가 커질수록 연산량의 감소율이 커진다는 것을 알 수 있었다. 두 번째 방법인 신호의 서브 프레임화를 통하여 FFT 내부의 연산량을 줄임으로써 연산량 감소를 유도하였다. 마지막으로 주파수 영역으로의 전환을 위해 QFT를 이용함으로써 곱의 연산량을 줄일 수 있었다. 제안된 방법 세 가지를 이용하여 음장효과를 구현한 결과, 수행 시간이 상당히 단축되어 실시간 구현의 가능성을 보여주었으며 룸 임펄스의 데이터를 손상시키지 않음으로써 보다 양질의 결과를 얻을 수 있었다. 그러나 QFT 크기가 어느 이상 커지게 되면 수행 시간이 반대로 증가하는 결과를 보였다. QFT 크기가 커질수록 QFT 수행 시간을 제외한 순수 연산 시간은 줄어드는 반면 QFT 수행 시간이 증가하기 때문이다.

참고 문헌

[1] H. V. Fuchs, X Zha, M. Spah, M. Prommer(1998), "Qualification of small free-filed and reverberation rooms for low frequencies", *Pro. of Conference Euro-Nosie 98*, pp. 657-662.

[2] J. Atagi (2002), "Effect of modulated delay time of reflection on autocorrelation function and perception of echo", *Journal of sound and vibration*, VOL. 258, pp. 443-450.

[3] Alan V. Oppenheim, Ronald W. Schafer (1999), "Discrete-time signal processing", Prentice Hall.

[4] Haitao Guo, Student Member (1998), "The Quick Fourier Transform: An FFT Based on Symmetries", *IEEE Transaction on signal processing*, VOL. 46, No. 2.

접수일자: 2004년 7월 10일
게재결정: 2004년 9월 9일

▶손성용(Song-Young Sohn)
주소: 305-732 대전광역시 유성구 문지로 119번지 한국정보통신대학교
소속: 한국정보통신대학원대학교(ICU) 음성/음향 정보 연구실
전화: 042)866-6296
E-mail: thill@icu.ac.kr

▶서정일(Jong-Il Seo)
주소: 305-732 대전광역시 유성구 가정동 161번지 한국전자통신연구원
소속: 한국전자통신연구원 전파방송연구소 방송 미디어 연구부
전화: 042)866-6206
E-mail: seoji@etri.re.kr

▶한민수(Minsoo Hahn)
주소: 305-732 대전광역시 유성구 문지로 119번지 한국정보통신대학교
소속: 한국정보통신대학원대학교(ICU) 음성/음향 정보 연구실
전화: 042)866-6123
E-mail: mshahn@icu.ac.kr