行政院國家科學委員會專題研究計畫 成果報告

# 快速音樂廳殘響設計
研究成果報告(精簡版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 94-2213-E-009-129-
執 行 期 間 ：94 年 08 月 01 日至 95 年 07 月 31 日
執 行 單 位 ：國立交通大學資訊工程學系(所)

計 畫 主 持 人：劉啟民

計畫參與人員：碩士級-專任助理：張子文
博士班研究生-兼任助理：楊宗翰

處 理 方 式 ：本計畫涉及專利或其他智慧財產權，2 年後可公開查詢

中 華 民 國 96 年 02 月 01 日

行政院國家科學委員會補助專題研究計畫 ■ 成 果 報 告
□期中進度報告

# 快速音樂廳殘響設計

計畫類別：■ 個別型計畫　　□ 整合型計畫
計畫編號：**NSC94-2213-E-009-129-**
執行期間：　94　　年　8　月　1　日至　95　年　7　月　31　日

計畫主持人：劉啟民
共同主持人：
計畫參與人員：張子文 楊宗瀚

成果報告類型(依經費核定清單規定繳交)：■精簡報告　□完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢

■涉及專利或其他智慧財產權，□一年■二年後可公開查詢

執行單位：　交通大學資訊工程系

中　華　民　國　九十六　年　一　月　二十　日

# 1 研究計畫之背景及目的

Artificial reverberators have been used to add reverberation to studio recording in the music and film industry, or to modify the acoustic of a listening room. There have been basically two approaches to design reverberators. The first approach is based on the IIR (Infinite Impulse Response)-recursive networks such as comb filters, all-pass filters. A variety of algorithms [9][10][11] have been proposed since the work of Schroeder [6][7]. The IIR-based network has the merit in low complexity, but is often difficult to eliminate unnatural resonances. On the other hand, the FIR (Finite Impulse Response) based reverberators, which convolve the input sequence with an impulse response modeling the concert hall, will be free from the unnatural resonances. However, the high computational complexity due to the long FIR length leads to another concern in real-time applications. For the two seconds of impulse response, the length will be 88,200 samples in terms of 44,100Hz sampling rate. Using direct convolution to implement the reverberation indicates the 88,200 multiplications for each sample, or 7.8G multiplications per second for stereo audio.

A lots of researches [1][4][17] have been developed to reduce the complexity of FIR-based reverberators. Among them, the FFT-based methods can significantly reduce the complexity. This project proposes a new idea in reducing complexity by combining the perceptual phenomenon with the FFT based method called the fast perceptual convolution. Besides, for having an effective quality measurement on the fast perceptual convolution, we examine the quality through an objective criterion which compares the perceptual difference between the tracks processed through the non-reduced FIR and the perceptually-reduced FIR. The result has shown a 30% improvement without affecting the perceptual reverberation quality. For more reduction, we also provide different reduction levels users' reference. In addition, we have also verified the quality of reverberation when using different reduction levels.

The objective of the project is to introduce the theoretical formulation on the FIR-based reverberation. This project will consider the fast algorithm based on the FFT method through the overlap-and-add method and the overlap-and-save method. We will derive the formula required. Also, we will consider the implementation of real-time FIR-based reverberation and the IIR-based reverberation for comparison. Then, this project intends to propose the perceptual convolution method. The method has been published in AES convention papers and the patents. However, there are some issues not investigated yet. This project intends to consider the issues. Another consideration is the applications to real-rtime ineracactive applications. In the application, the delay or the latency need to be very short. This project extends the result and derivation for the low-delay perceptual convolution. The forth objective of the project is on the real-time demonstration system. The project intends to put into the realization through several reverberators. The real-time system will analyz the computing complexity, memory required, and the computing speeds. The fifth topic is on the objective and subjective test measurement to prove the quality of the reverberators. We collect the reverations and consider the test data base used to test the reverberators.

## 1.1 Reverberation

Reverberation is a complicated echo system. The listener in a room hears not only the direct signal from the source, but also other reflected sounds from the walls, floor or some other objects in the room. As shown in *Figure 1.1*, the signal heard by the listener is a summation of all reflected signals.

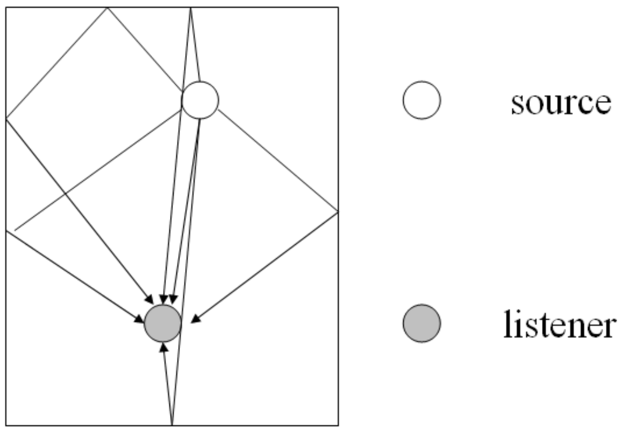

*Figure 1.1: Reverberation.*

The effect of reverberation is a multiplicity echoes placed very close that are not perceptually separate from one another. *Figure 1.2* shows the impulse response of the Foellinger Great Hall. From *Figure 1.2*, we can see that the peaks for later part of the impulse response are very close, only few peaks in the earlier part are clearly stood out of the response. By this characteristic, the reverberation can be separated into two parts. As shown in *Figure 1.3*, those peaks in earlier part were called earlier reflections, and the later part is called late reverberation.

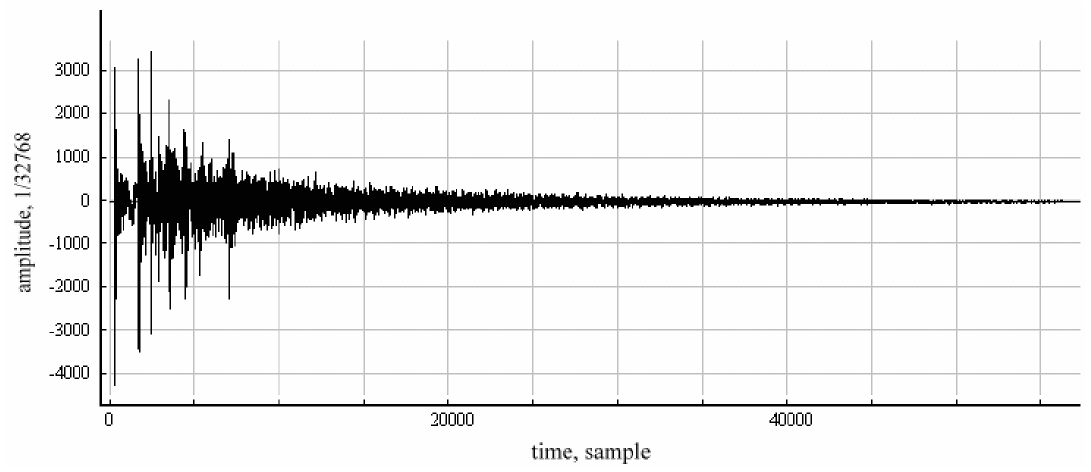

*Figure 1.2: Impulse response of Foellinger Great Hall (Sonic Foundry)*

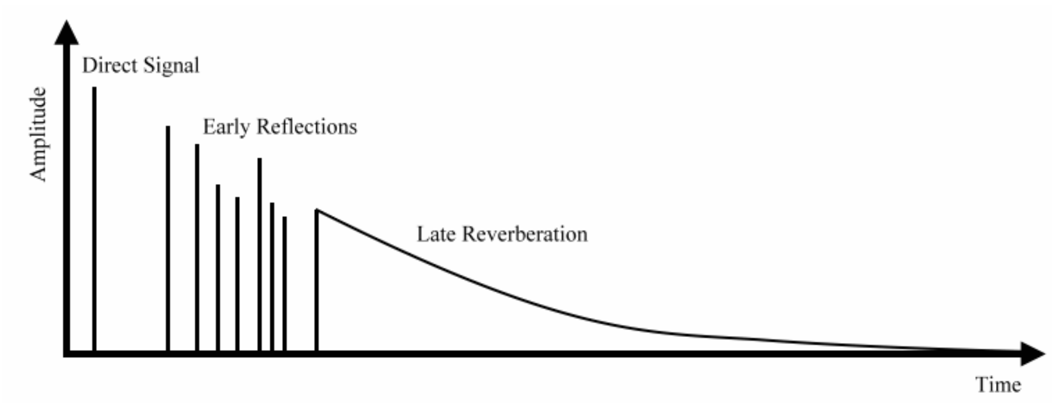

*Figure 1.3: Early reflections and late reverberation*

## 1.2 FIR-based Approach and IIR-based Approach

Artificial reverberation can be implemented by two approaches. The first one [4][17] is through the convolution of the impulse response and source signals which is referred to as the FIR-based approach. The second one [6][7][8][9][11] combines the various filters like all-pass filter, comb filters, and FIR filters to establish the reverberation effect, which is referred to as the IIR-based approach. The first approach usually leads to a better effect with higher computing complexity compared to the second approach. There are some researches [15] trying to take the advantages of both approaches by developing hybrid algorithms. This section will introduce those approaches.

This FIR-based approach records the environment response, such as a concert hall or a church, as the impulse response and then applies the direct convolution to have the reverberation effect. The environment response can be recorded from real environments using a loud speaker and microphones. *Figure 1.2* is an example of environment response. The length of a natural environment response might be varying from 1 to several seconds depending on the size of the room, the material of the walls and other surfaces in the room. For 2 seconds of impulse response, the length will be 88,200 samples in terms of 44,100Hz sampling rate. By direct convolution, convolving a stereo input signal with such impulse response needs 7.8G multiplications per second.

This is almost impossible for processors today. Section 1.3 will introduce some techniques to reduce the complexity of convolution for very long impulse response.

The IIR-based approach suitably combines various filter modules such as comb filters, all-pass filters, and low-pass filters to simulate the reverberation effect. Due to the nature of the recursive filters, the complexity is in general lower than the FIR-based approach. However, the quality needs some detail calibration and also it will be difficult to model the existing environment directly. Section 1.4 introduces the IIR filters and some IIR-based reverberators made by those filters.

## 1.3 FIR-based Approach

Physical approach can be implemented by convolution methods. This section will introduce the operation of convolution and the block convolution method for FFT convolution to reduce its complexity.

### 1.3.1 Direct Convolution

The convolution between input signal $x[n]$ and impulse response $h[n]$ of length $L$ is expressed as

$$y[n] = x[n] * h[n] = \sum_{k=0}^{L-1} x[n-k]h[k] \qquad (1)$$

The direct implementation of (1) is shown in *Figure 1.4*. This implementation leads to $L$ multiplications per output sample, which is too complicated for reverberation. A much more efficient method is to compute convolution through the block convolution, in which the signal and impulse response is segmented into sections of length $N$. Convolution of each block convolution is then implemented through the Fast Fourier Transform (FFT).

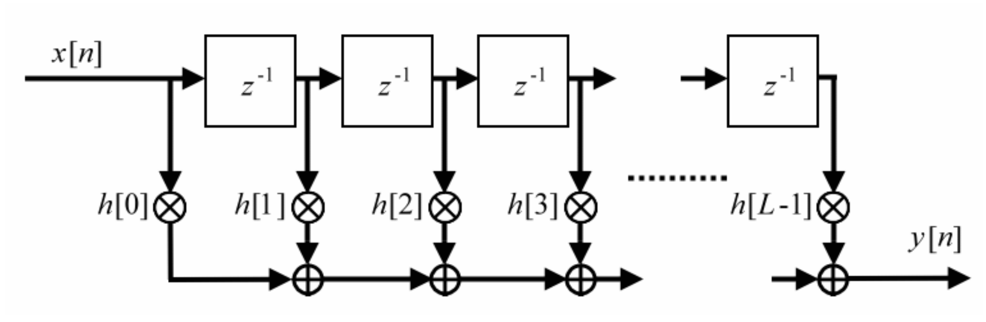

*Figure 1.4: Block diagram of direct convolution*

### 1.3.2 Block Convolution

Because we need to process segmented input signal, methods to recombine the processed segments into final signal are needed. There have been two approaches: overlap-and-save [14] method and overlap-and-add [16] method.

For overlap-and-add method, we will do the convolution on each input segment. If the input segment size is $N$ and the impulse response length is $L$, it will produce $N+L-1$ samples of output for each segment. The later $L-1$ samples of each output segment will affect its following output segments. For example, let us consider about the signals shown in *Figure 1.5*.

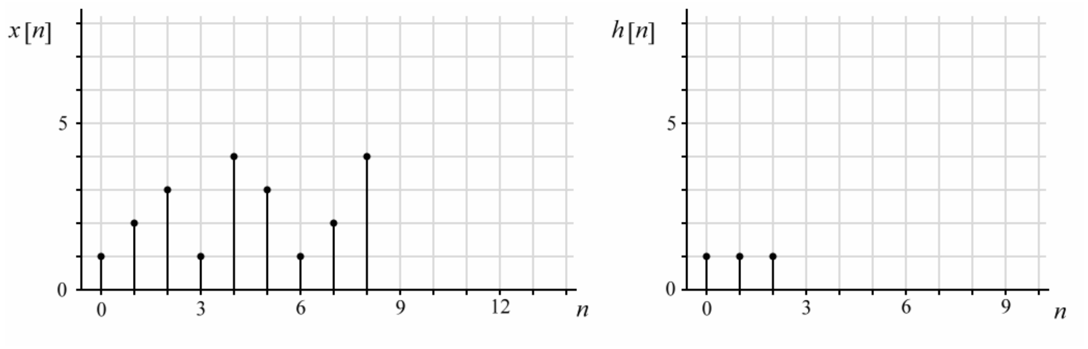

*Figure 1.5: Overlap-and-add example (input signal x[n] and impulse response h[n])*

The length of the input signal $x[n]$ is 9 and the length of the impulse response $h[n]$ is 3. As shown in *Figure 1.6*, if we choose the input block size of 3, the input signal will be separated into

3 blocks. For each small block $x_r[n]$, we do convolution to produce the corresponding output $y_r[n]$. Then, we add those output blocks to produce the result signal $y[n]$. This result is equivalent with the result produced by direct convolution.

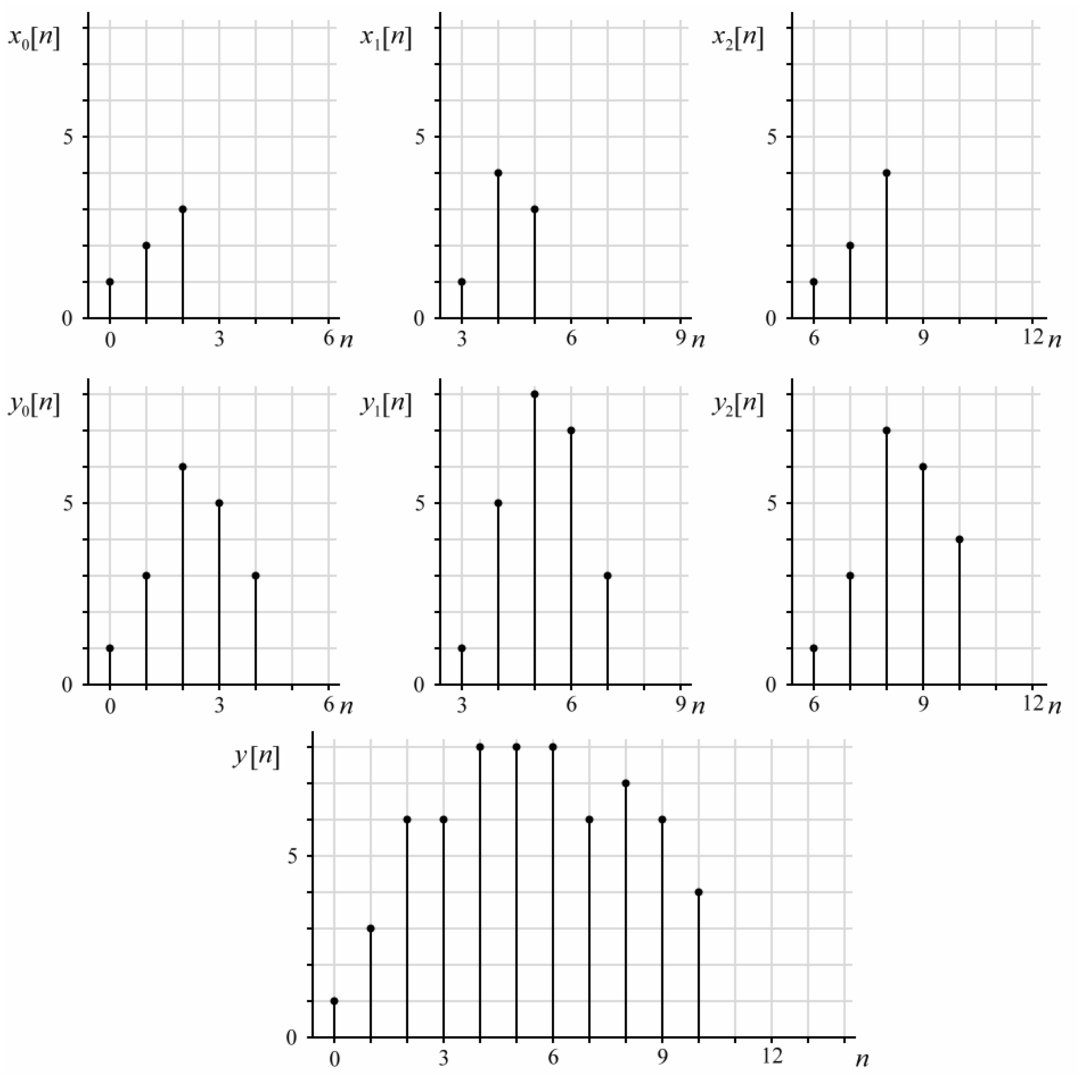

*Figure 1.6: Overlap-and-add example (input blocks $x_r[n]$, output blocks $y_r[n]$, and the final output result $y[n]$)*

To extend the overlap-and-add approach to segmented impulse response, let the input signals $x[n]$ and impulse response $h[n]$ are segmented as a sum of shifted finite-length segments of length $N$; i.e.,

$$x[n] = \sum_{r=0}^{\infty} x_r[n - rN], \tag{2}$$

and

$$h[n] = \sum_{s=0}^{M-1} h_s[n - sN], \tag{3}$$

where $M$ is the smallest integer larger than $L$ divided by $N$, i.e. $M = \left\lceil \frac{L}{N} \right\rceil$

$$x_r[n] = \begin{cases} x[n + rN], & 0 \le n \le N - 1 \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

and

$$h_s[n] = \begin{cases} h[n + sN], & 0 \le n \le N - 1 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Substituting (2) and (3) into (1) yields

$$y[n] = \left\{ \sum_{r=0}^{\infty} x_r[n - rN] \right\} * \left\{ \sum_{s=0}^{M-1} h_s[n - sN] \right\} \tag{6}$$

Because convolution is linear time-invariant, it follows that

$$y[n] = \sum_{r=0}^{\infty} \sum_{s=0}^{M-1} x_r[n - rN] * h_s[n - sN] = \sum_{r=0}^{\infty} \sum_{s=0}^{M-1} y_{r,s}[n - rN - sN], \tag{7}$$

where

$$y_{r,s}[n] = x_r[n] * h_s[n] \qquad \text{for } 0 \le n < 2N - 1 \tag{8}$$

The overlap-and-save method is very similar to the overlap-and-add except the input blocks are overlapped, and the output blocks are not overlapped. By overlap-and-save method, when the input block size is $N$, for each input block, it will combined with previous $L-1$ samples to a new block with $N+L-1$ samples. Then we perform circular convolution or linear convolution on each input block. The first $L-1$ samples of each output block are discarded. If linear convolution is used, the tailing $L-1$ samples of each output block are also discarded. Finally, the output blocks are concatenated to form the result output.

Consider the example used in overlap-and-add method. As shown in *Figure 1.7*, the input blocks $x_r$[n] were selected in length of 5 including previous 3−1=2 samples. Then perform 5-point circular convolution on each input block to produce the corresponding output $y_r[n]$. Then, the first 2 samples of each output block are discarded, and concatenated to produce the result signal $y[n]$.

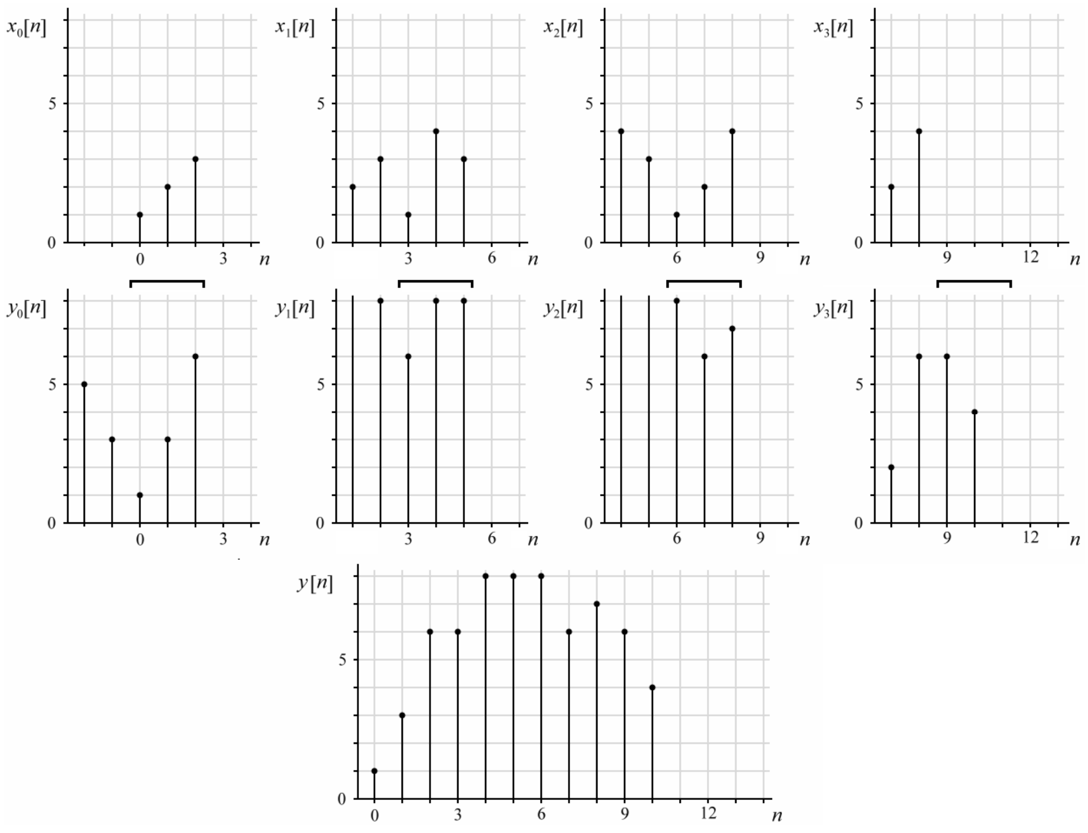

*Figure 1.7: Overlap-and-save example (input blocks $x_r[n]$, output blocks $y_r[n]$, and the final output result $y[n]$)*

To extend overlap-and-save method to segmented impulse response, we begin by changing the parameter $r' = r + s$ in (7):

$$y[n] = \sum_{r'=0}^{\infty} \sum_{s=0}^{M-1} y_{r'-s,s}[n - r'N] , \tag{9}$$

Define

$$y'_{r'}[n - r'N] = \sum_{s=0}^{M-1} y_{r'-s,s}[n - r'N] , \tag{10}$$

where

$$y_{r'-s,s}[n] = x_{r'-s}[n] * h_s[n] \qquad \text{for } 0 \le n < 2N - 1 \ . \tag{11}$$

(9) can be represented as

$$y[n] = \sum_{r'=0}^{\infty} y'_{r'}[n - r'N] , \tag{12}$$

$y'_{r'}[n - r'N]$ is the summation of all blocks in time interval $[r'N, (r'+2)N-1]$. The form required in the overlap-and-save should be to separate the output into $y_r[n]$ be the non-overlapping blocks; that is,

$$y[n] = \sum_{p=0}^{\infty} y_p[n - pN] \tag{13}$$

where

$$y_p[n] = \begin{cases} y[n + pN], & 0 \le n \le N - 1 \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

Substituting (12) into (14) yields

$$y_p[n] = \sum_{r'=0}^{\infty} y'_{r'}[n + pN - r'N], \qquad 0 \le n \le N - 1 \tag{15}$$

Since each $y_{r'}[n - pN - r'N]$ represents the values at time interval $2N$, there is only two terms in the intervals $[0, N-1]$; that is

$$y_p[n] = y'_{p-1}[n+N] + y'_p[n], \qquad 0 \le n \le N-1 \tag{16}$$

Substituting (10) and (11) into (16) yields

$$y_p[n] = \sum_{s=0}^{M-1} x_{p-s-1}[n+N] * h_s[n] + \sum_{s=0}^{M-1} x_{p-s}[n] * h_s[n], \qquad 0 \le n \le N-1 \tag{17}$$

$$= \sum_{s=0}^{M-1} \left\{ x_{p-s-1}[n+N] + x_{p-s}[n] \right\} * h_s[n] \qquad 0 \le n \le N-1 \tag{18}$$

Let

$$x'_p[n] = x_{p-1}[n+N] + x_p[n], \qquad -N \le n \le N-1 \tag{19}$$

where $x'_p[n]$ is $p$-th overlapping block of the input signal $x[n]$. Then, (18) can be rewritten as

$$y_p[n] = \sum_{s=0}^{M-1} x'_{p-s}[n] * h_s[n], \qquad 0 \le n \le N-1 \tag{20}$$

Form (20), each non-overlapping output block can be calculated by evaluating the convolution for overlapping input blocks in the corresponding time interval.

In overlap-and-add or overlap-and-save, the convolution of each pair of small blocks can be transform to DFT domain and perform multiplications on DFT domain. Since the complexity of specific sizes of DFT can be reduced from O($n^2$) to O($n$log$n$) by FFT algorithms. Using these algorithm to perform the convolution can significant reduce the complexity. The method and complexity of FFT is given.

### 1.4 IIR-Based Approach

Using methods in FIR-based approach for reverberation may require massive computing power. Although, extremely accurate simulation is necessary for some applications (such as echo cancellation), such accuracy is not necessary to achieve a convincing artificial reverberation effect.

Perceptual approach is to realize real-time artificial reverberation that is perceptively indistinguishable from real reverberation. Most reverberation algorithms in perceptual approach are implemented by combining some small DSP blocks such as inverse comb filters, comb filters, all-pass filters and low-pass filters. Different combinations are attempted to simulate the reverberation effect of various rooms. This section will introduce these filters and their characteristics. The reverberation of this approach is trying to match the general characteristics of the impulse response for natural environments, such as exponentially decay late reverberation as shown in *Figure 1.3*.

### 1.4.1 Inverse Comb Filter

The inverse comb filter is to produce one echo to the input signal. This can be accomplished by adding a feed-forward path with delay to the signal path. The block diagram of inverse comb filter is illustrated in *Figure 1.8*. The absolute value of the gain *g* in the feed-forward path is a coefficient less than 1.

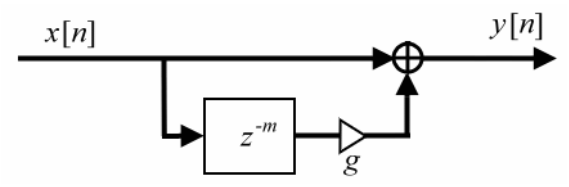

*Figure 1.8: Block diagram of inverse comb filter*

The difference equation of inverse comb filter is as follow:

$$y[n] = x[n] + gx[n-m] \qquad (21)$$

The *z*-transform is

$$H(z) = 1 + gz^{-m} \qquad (22)$$

and the impulse response is

$$h[n] = \delta[n] + g\delta[n-m] \tag{23}$$

Because the impulse response length is finite, the filter is a FIR filter. From (22), we can derive the frequency response:

$$H(e^{j\hat{\omega}}) = 1 + ge^{-jm\hat{\omega}} \tag{24}$$

where $\hat{\omega}$ is the normalized frequency. The magnitude response is given by

$$\begin{aligned} \left|H(e^{j\hat{\omega}})\right| &= \left|1 + g\cos(m\hat{\omega}) - gj\sin(m\hat{\omega})\right| \\ &= \sqrt{[1 + g\cos(m\hat{\omega})]^2 + g^2\sin^2(m\hat{\omega})} \end{aligned} \tag{25}$$

Then, we can plot the magnitude response of inverse comb filter as shown in *Figure 1.9*.

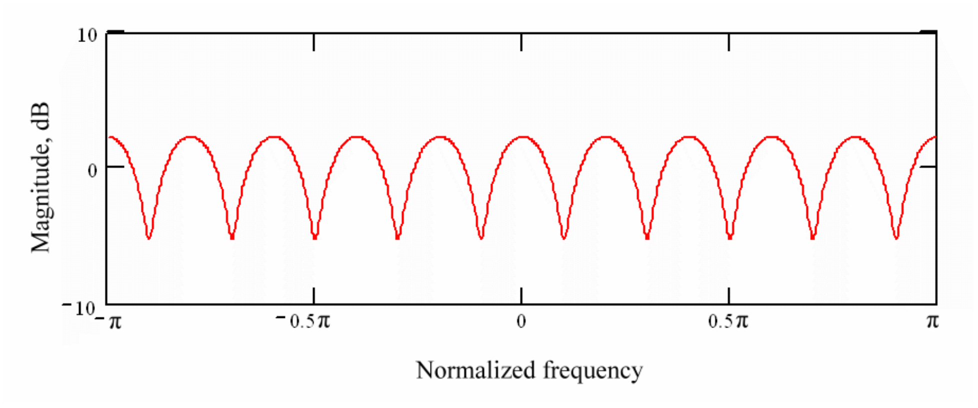

*Figure 1.9: Magnitude response of inverse comb filter (m=10, g=0.7)*

Because the inverse comb filter will only produce one echo, using this filter in reverberation designs, the density of echoes is not dense enough. In practical designs, we usually use the comb filter instead of inverse comb filter.

### 1.4.2 Comb Filter

The comb filter is to produce echoes with feedbacks. The echoes produced will be used as the input to produce their echoes. Hence, this kind of filters should be IIR filters. This filter can be implemented by adding a feed-backward path with delay to the signal path. The block diagram of comb filter is shown in *Figure 1.10*. The absolute value of the gain *g* should be less than 1 to make the system stable.

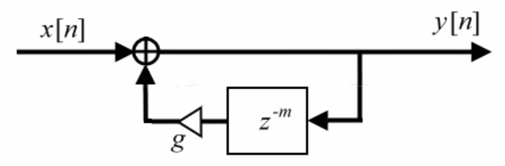

*Figure 1.10: Block diagram of comb filter*

This filter can work alone to be a reverberator in some low requirement applications, such as the "echo" effect used in many applications. In other reverberation designs, they will not need to have the direct signal, because the comb filters may be placed in parallel. The comb filter will be modified to the one shown in *Figure 1.11* to be more suitable for reverberation designs. The one shown in *Figure 1.11* is similar to the one shown in *Figure 1.10*, however, the delay line and attenuation gain of this design are located at the direct path. Note that the impulse response produced by these two designs will be similar to each other, but one is smaller than the other by a factor of *g*, and delayed *m* samples.

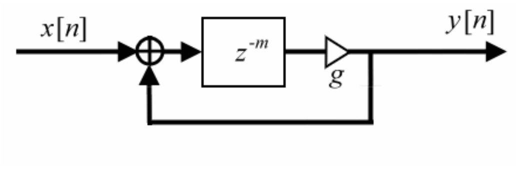

*Figure 1.11: Block diagram of comb filter (delayed)*

The difference equation of the comb filter is shown as follows:

$$y[n] = gx[n-m] + gy[n-m] \qquad (26)$$

The *z*-transform of the comb filter is

$$H(z) = \frac{gz^{-m}}{1 - gz^{-m}} \tag{27}$$

and its impulse response can be expressed as

$$\begin{aligned} h[n] &= g\delta[n-m] + g^2\delta[n-2m] + g^3\delta[n-3m] + \cdots \\ &= \sum_{k=1}^{\infty} g^k \delta[n-km] \end{aligned} \tag{28}$$

Hence, the frequency response of the comb filter can be expressed as

$$H(e^{j\hat{\omega}}) = \frac{ge^{-jm\hat{\omega}}}{1 - ge^{-jm\hat{\omega}}} \tag{29}$$

The magnitude response of the comb filter is shown in *Figure 1.12*.

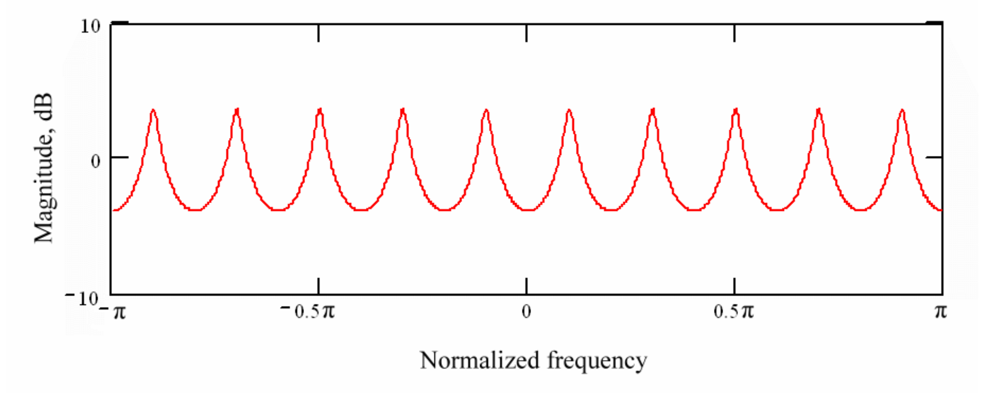

*Figure 1.12: Magnitude response of comb filter (m=10, g=0.7)*

The reverberation time $T_{60}$ (time to decay 60dB) of comb filter is given by

$$T_{60} = \frac{60}{-20\log_{10}(g)} mT \tag{30}$$

where $T$ is the sampling period. For given delay $m$ and reverberation time $T_{60}$, the attenuation gain $g$ can be evaluated as

$$g = 10^{\frac{-3mT}{T_{60}}} \tag{31}$$

When the echo density of comb filter is not enough, it causes fluttering sound on transient inputs. Reducing the delay length *m* can increase the echo density. However, from *Figure 1.12*, there are *m*/2 frequency peaks between 0 to π. Reducing the delay length will also decrease the number of the peaks in frequency domain. This will make a sound that resonates at specific frequencies.

### 1.4.3 All-Pass Filter

To avoid resonation of comb filter, Schroeder [6] suggested to use the all-pass filter which adds a feed-forward path to the comb filter. The block diagrams of two all-pass filters are shown in *Figure 1.13* and *Figure 1.14*. Although, some will use the one shown in *Figure 1.14* instead of the one shown in *Figure 1.13*, the properties of both designs are equivalent.

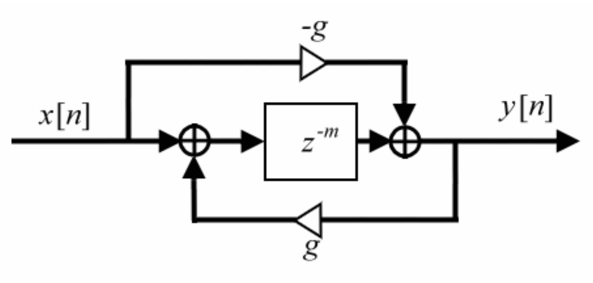

*Figure 1.13: Block diagram of all-pass filter*

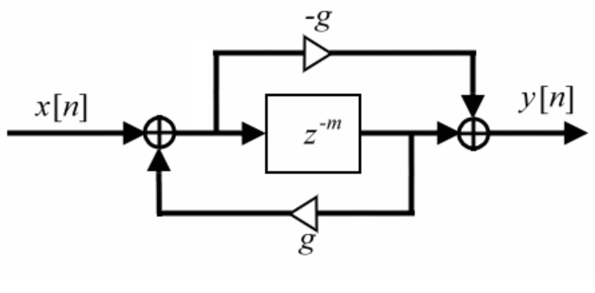

*Figure 1.14: Block diagram of all-pass filter (different version)*

The difference equation of all-pass filter is given by

$$y[n] = -gx[n] + x[n-m] + gy[n-m] \tag{32}$$

Hence, its *z*-transform is

$$H(z) = \frac{-g + z^{-m}}{1 - gz^{-m}} \tag{33}$$

and its impulse response is expressed as

$$\begin{aligned} h[n] &= -g\delta[n] + \delta[n-m] - g^2\delta[n-m] + g\delta[n-2m] - g^3\delta[n-2m] + \cdots \\ &= -g\delta[n] + (1-g^2)\delta[n-m] + g(1-g^2)\delta[n-2m] + \cdots \\ &= -g\delta[n] + (1-g^2)\sum_{k=1}^{\infty} g^{k-1}\delta[n-km] \end{aligned} \tag{34}$$

The frequency response of all-pass filter is given by

$$H(e^{j\hat{\omega}}) = \frac{-g + e^{-jm\hat{\omega}}}{1 - ge^{-jm\hat{\omega}}} \tag{35}$$

$$= e^{-jm\hat{\omega}} \frac{1 - ge^{jm\hat{\omega}}}{1 - ge^{-jm\hat{\omega}}} \tag{36}$$

From (36), we found that the magnitude of $e^{-jm\omega}$ is 1 for all $\omega$, the magnitude of the quotient of complex conjugates is also 1. Therefore, the frequency response of all-pass filter is unity. Hence,

$$\left|H(e^{j\hat{\omega}})\right| = 1 \tag{37}$$

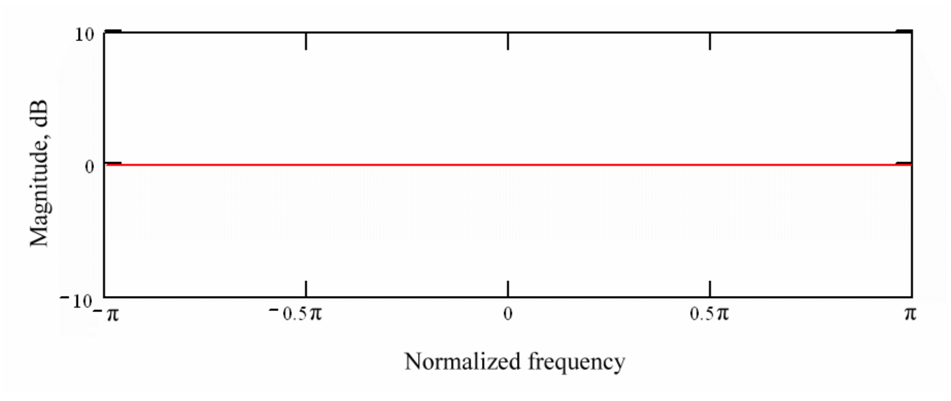

*Figure 1.15: Magnitude response of all-pass filter*

From (34), as compared with (28), the impulse responses of all-pass filter and comb filter are similar to each other. Except the first pulse, the impulse response of all-pass filter is smaller than that of the comb filter by a factor of $(1-g^2)$. Because of this property, outputs of both filters sound similar. The resonation effect like comb filter does can still be heard. This is because the flat frequency response can be true only when the analysis window size is big enough. However, the perceptive window size is limited in short period.

The complexity of IIR filters can be calculated by summing up the complexity of its blocks. The number of multiplications needed per sample for comb filters and all-pass filters is 1 and 2, respectively.

### 1.4.4 Reverberation Filters

Using comb filter or all-pass filter alone may not be able to increase the density of echoes and the density frequency peaks together. However, it can be accomplished by combining those blocks together. To increase the density of echoes, Schroeder cascaded all-pass filters as shown in *Figure 1.16*. The frequency response of this reverberator is also all-pass, since the all-pass filter is a linear time-invariant system.

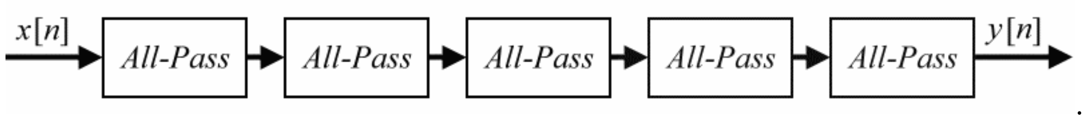

*Figure 1.16: Schroeder's series all-pass reverberator.*

By this combination, the echoes generated by the first all-pass filter will be used to generate more echoes in next all-pass filter. But the reverberation generated by this reverberator will sound unnatural, especially when the input is transient.

Instead of using all-pass filters, Schroeder suggested combining comb filters in parallel and cascade all-pass filters to give the reverberator shown in *Figure 1.17*.

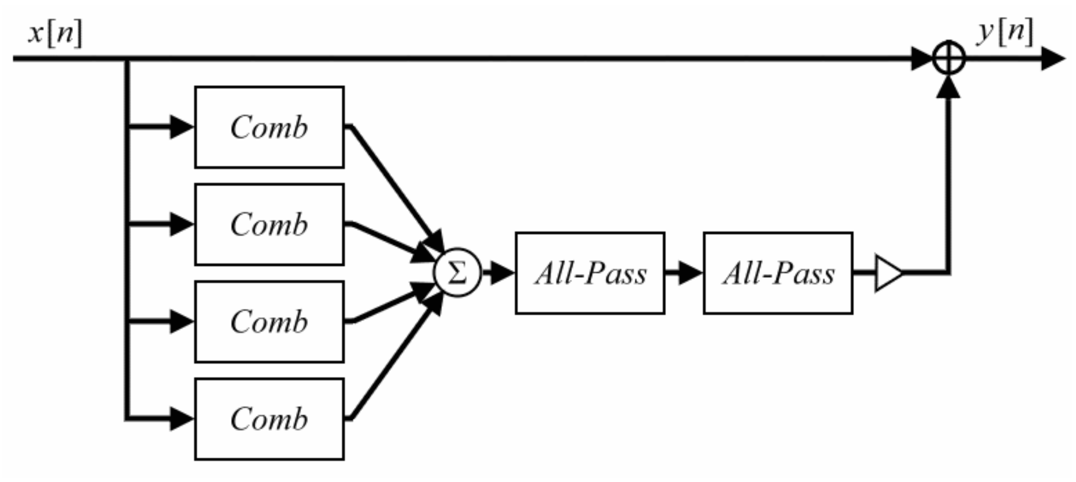

*Figure 1.17: Schroeder's reverberator*

The delays in comb filters are chosen to be relatively primes to avoid overlapping their peaks in frequency response, and the attenuation gains are chosen to have the same reverberation time. From (31), yields

$$\gamma = g_p^{1/m_p} \qquad \text{for any } p \tag{38}$$

where

$$\gamma = 10^{-3T/T_{60}} \tag{39}$$

$g_p$ and $m_p$ are the attenuation gain and delay length of the $p$-th comb filter, respectively. The cascaded all-pass filters are used to increase the echo density without modifying the frequency response. The reverberator provides more natural reverberation as compared to the series all-pass reverberator.

Since each comb filter needs one multiplication per input sample, and each all-pass filter needs two multiplications per input sample, the number of multiplications needed by Schroeder's reverberator is 4×1+2×2+1=9.

Although, the voice produced by Schroeder's reverberator is still far from the natural ones.

Comparing to the impulse response of a natural environment shown in *Figure 2.5*, the energy of higher frequency part will decay faster than that of the lower frequency part. It is because the energy of higher frequencies in natural environment will be absorbed by air, walls or other objects in the room. To solve this problem, Moorer [8] proposed a reverberator shown in *Figure 1.18*. The reverberator looks similar to Schroeder's. It is combined with 6 parallel comb filters and cascaded with one all-pass filter. The major difference is that Moorer inserted a first order IIR low-pass filter in the feedback path of each comb filter to simulate the environment absorption. The absorbent comb filter is shown in *Figure 1.19*. This makes the reverberation time a function of frequency (shorter reverberation time for higher frequencies).

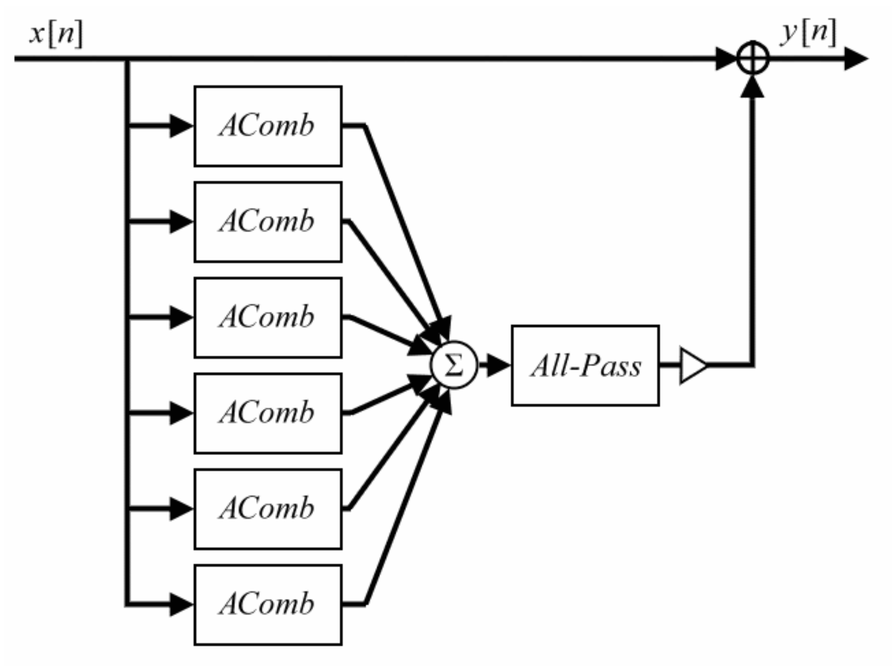

*Figure 1.18: Moorer's reverberator*

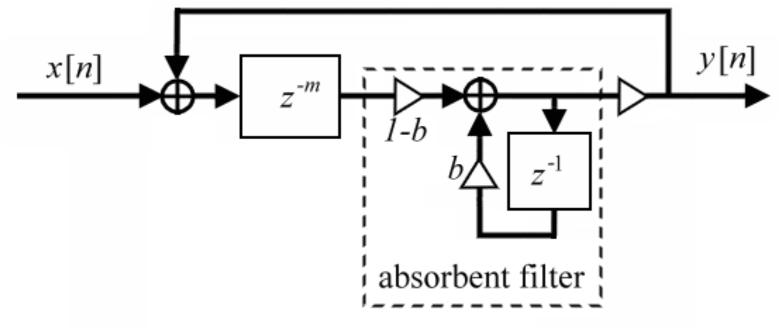

*Figure 1.19: Absorbent comb filter used in Moorer's reverberator*

Since the absorbent filter needs two multiplications per sample, each absorbent comb filter needs three multiplications per sample. Therefore, the number of multiplications needed by Moorer's reverberator is 6×3+2×1+1=21.

The reverberation generated by Moorer's reverberator still sounds fluttering and metallic on transient inputs. Jot [9] pointed out that it is difficult to obtain a sufficient time density with a reasonable number of unit filters, given that the total delay length determine s the maximum frequency density one can obtain. Jot tried to using general delay network as shown in *Figure 1.20* to improve the echo density with a small number of delay units.

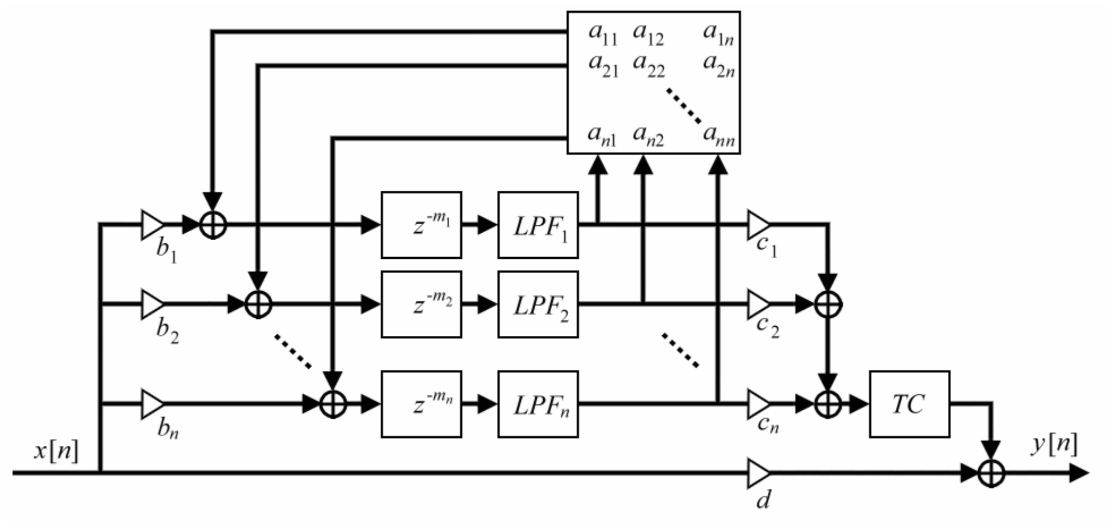

*Figure 1.20: Jot's general delay network*

Unlike pervious reverberators, the feedback of each delay unit in general delay will feed to other delay units. This property helps to generate higher echo density, and removes resonation of the original parallel comb filters. As in Moorer's reverberator, low-pass filters can be inserted to obtain frequency dependent reverberation time. The number of multiplications per sample for $n$-level general delay network is $3n+n^2+2$. For 5-level general delay network, it needs 42 multiplications per sample.

## 1.5 Hybrid Reverberators

Besides the reverberators in those two approaches introduced in previous sections, Browne [15] raised a hybrid algorithm that combined a truncated impulse response convolution and a IIR-based reverberator to tradeoff the two approaches. The block diagram of this algorithm is shown in *Figure 1.21*. The convolution phase uses only the earlier reflection part of the impulse response which is about 50ms to 150ms. This can significantly reduce the convolution filter length. The block convolution phase is implemented with 8192-point FFT without partitioning the impulse response. The recursive filter is implemented by using Moorer's design to provide frequency dependent reverberation time.

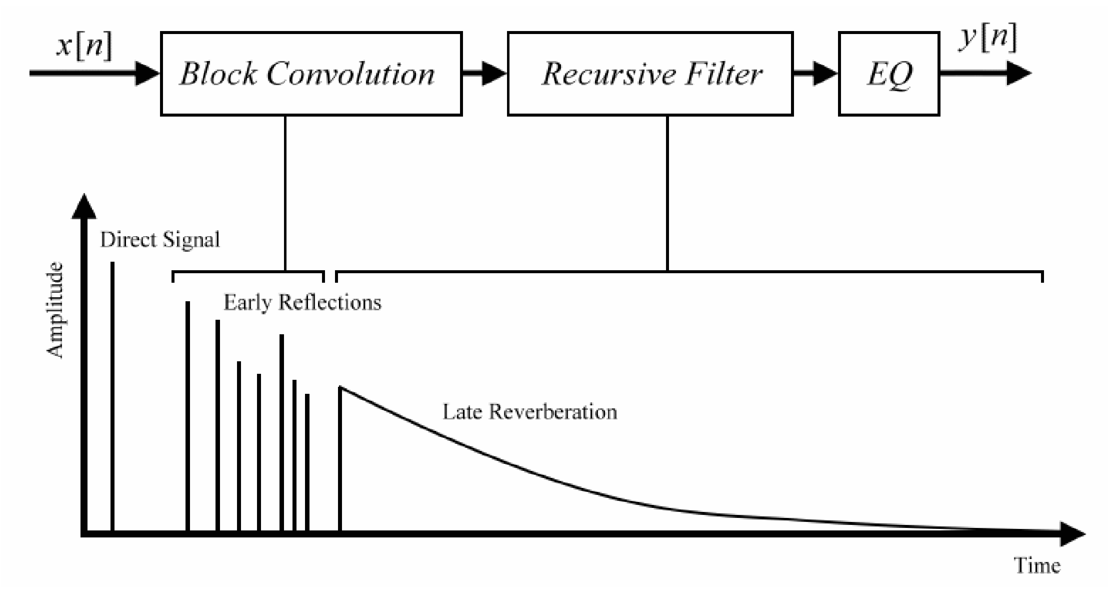

*Figure 1.21: Hybrid reverberator proposed by Browne[15]*

# 2 研究方法、進行步驟及執行進度

## 2.1 Block Convolution Performed through FFT

From section 1.3.2, we discussed that the linear convolution of a long impulse response, we can separate both input signal $x[n]$ and impulse response $h[n]$ into blocks. The convolution each pair of input signal block $x_r[n]$ and impulse response block $h_s[n]$ can be implemented with the FFT with $2N-1$ points. We adopt for complexity evaluation based on radix-2 FFT and $2N$-point FFT instead of $(2N-1)$-point FFT. Let

$$\hat{x}_r[n] = \begin{cases} x[n+rN], & 0 \le n \le N-1 \\ 0, & N-1 < n \le 2N-1 \end{cases} , \tag{40}$$

and

$$\hat{h}_s[n] = \begin{cases} h[n+sN], & 0 \le n \le N-1 \\ 0, & N-1 < n \le 2N-1 \end{cases} , \tag{41}$$

Since the convolution in time domain refers to the multiplication in frequency domain, (8) can be written as

$$Y_{r,s}[k] = X_r[k] \cdot H_s[k]; \quad for\ 0 \le k < 2N \tag{42}$$

where $Y_{r,s}[k]$, $X_r[k]$, and $H_s[k]$ are the $2N$-point FFT of $y_{r,s}[n]$, $\hat{x}_r[n]$ and $\hat{h}_s[n]$, respectively. According to the above derivation, we can summarize a fast algorithm as Algorithm 1 shown in the following.

Step 1: Store the FFT data of the segmented impulse response, $H_s[k]$.

Step 2: Execute $2N$-point FFT on the segmented input signals to obtain $X_r[k]$.

Step 3: Multiply $M$ pairs of FFT data according to (42). The number of multiplications and additions for each input sample are $2M$ and 0, respectively. Because the input signal and the impulse response are both real signals, the negative frequency part data will be the complex conjugate of the positive frequency part. By this property, we can calculate only $N+1$ multiplications for each block. This will reduce the number of multiplications for each input sample to $M+M/N$.

Step 4: Perform $M$ times the inverse FFT to have the segmented data $y_{r,s}[n]$ for different $s$.

Step 5: Overlap and add all the segmented $y_{r,s}[n]$ to have the final $y[n]$ according to (7). The number of additions is $2(M-1)$ for each input sample.

The number of complex multiplications needed per input sample is $(1+M)\mathrm{FFT}(2N)/N+M+M/N = (1+M)(\log_2 N+1)/2-1/N+M$. The algorithm has reduced the complexity of multiplications from $L$ to $2(1+M)(\log_2 N+1)-4/N+4M$. The block diagram for this algorithm is shown in *Figure 2.1*.

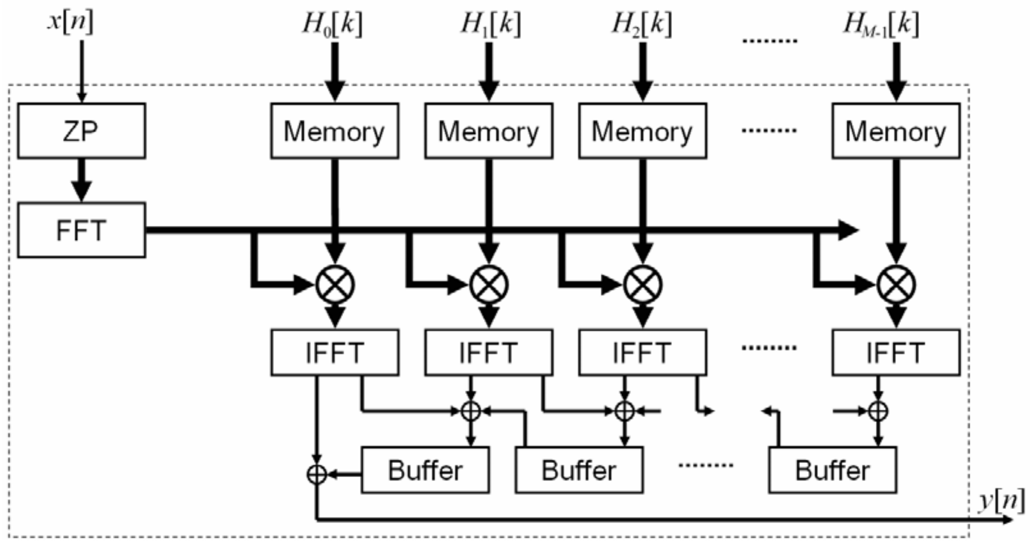

*Figure 2.1: Block diagram of FFT convolution (Algorithm 1)*

To reduce the complexity of Algorithm 1, we can change the order of calculations in

Algorithm 1. Let $p$=$r$+$s$, (7) is rewritten as

$$y[n]=\sum_{p=s}^{\infty}\sum_{s=0}^{M-1}y_{p-s,s}[n-pN]=\sum_{p=s}^{\infty}\sum_{s=0}^{M-1}x_{p-s}[n-(p-s)N]*h_s[n-sN]\,. \tag{43}$$

Define

$$y_p[n]=\sum_{s=0}^{M-1}y_{p-s,s}[n-pN]=\sum_{s=0}^{M-1}x_{p-s}[n-(p-s)N]*h_s[n-sN] \tag{44}$$

Hence,

$$y[n]=\sum_{p=s}^{\infty}y_p[n] \tag{45}$$

The nonzero values of $y_p[n]$ is only in the time interval [$pN$, $pN$+2$N$−2]. Let $n'=n-pN$, we have

$$y_p[n'+pN]=\sum_{s=0}^{M-1}y_{p-s,s}[n'] \tag{46}$$

Performing 2$N$-point FFT on (46) within the nonzero interval [0, 2$N$−1] leads to

$$Y_p[k]=\sum_{s=0}^{M-1}Y_{p-s,s}[k]=\sum_{s=0}^{M-1}X_{p-s}[k]H_s[k] \qquad \text{for } 0\le k<2N-1 \tag{47}$$

The fast convolution, denoted as Algorithm 2, is summarized as follows.

Step 1: Store the FFT data of the segmented impulse response, $H_s[k]$.

Step 2: Execute 2$N$-FFT on the segmented input signals to obtain $X_r[k]$.

Step 3: Multiply and add the two FFT data according to (47). The number of multiplications and additions is both $M$+$M$/$N$ for each input sample.

Step 4: Perform inverse FFT to have the segmented data $y_p[n]$.

Step 5: Overlap and add all the segmented $y_p[n]$ to have the final $y[n]$ according to (45). The overlapping factor is 1 and hence has the complexity one.

The block diagram of the fast convolution is illustrated in *Figure 2.2*. The complexity of multiplications in Algorithm 2 is 2FFT(2$N$)/$N$+$M$+$M$/$N$, which has a factor up to $M$ times reduction than Algorithm 1.

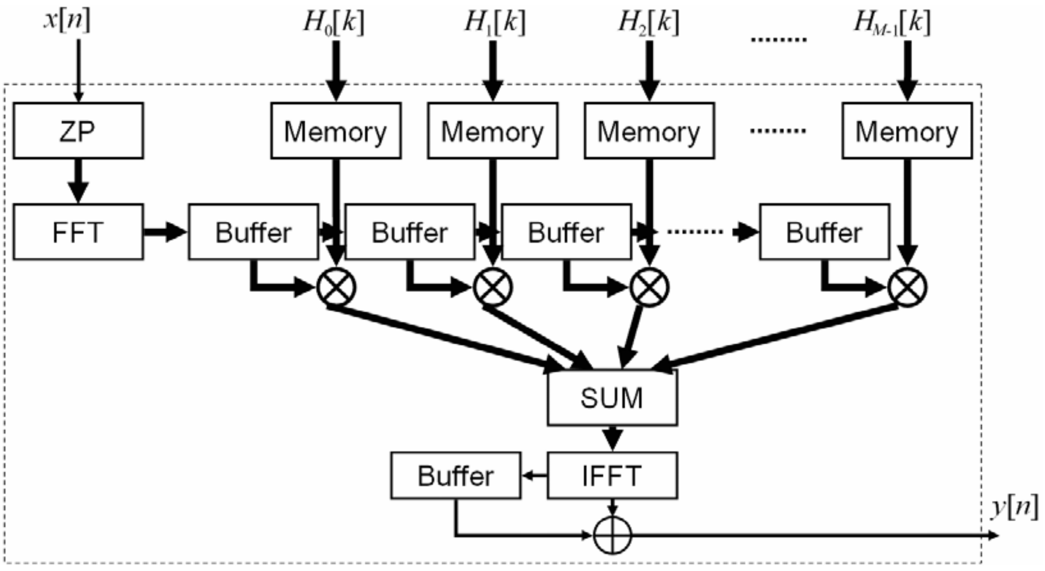

*Figure 2.2: Block diagram of FFT convolution (Algorithm 2)*

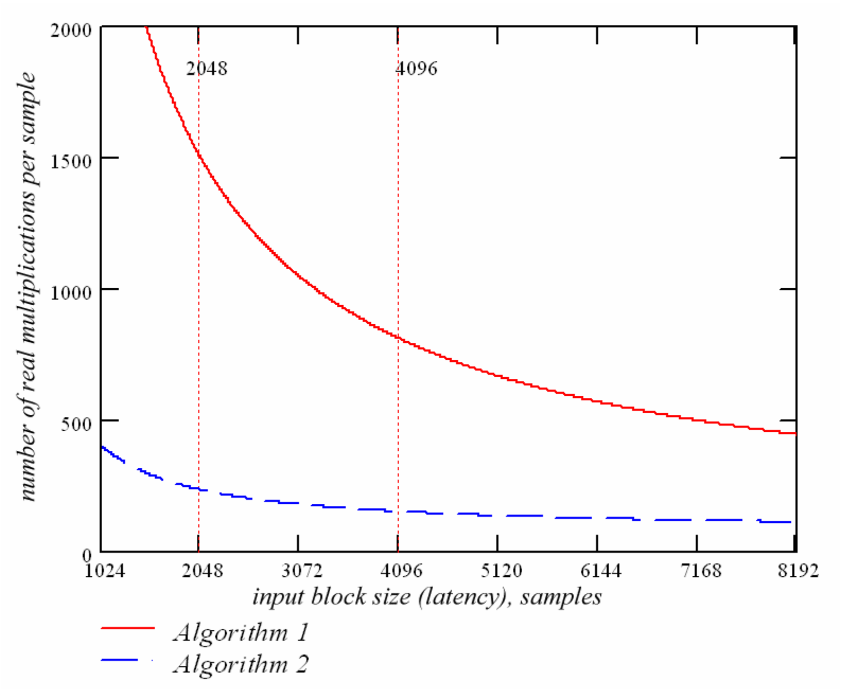

*Figure 2.3: Comparison of Algorithm 1 and Algorithm 2 when impulse response length is 2 seconds (88,200 samples)*

*Figure 2.3* illustrates Algorithm 1 and Algorithm 2 in the number of real multiplications per sample. When the input block size is set to 4096, Algorithm 2 needs about 150 real multiplications to convolve a signal with 88,200 samples of impulse response.

## 2.2 Block Size Analysis

Since the block size is the latency of the system, we will try to shorten the block size to reduce the latency of the system, though shortening the block size will increase the complexity of the system. For efficiency, we tried to increase the block size in an acceptable range to reduce the complexity. The acceptable latency in applications is about 150 ms, about 6K samples in terms of 44,100 Hz sampling rate. From *Figure 2.3*, the number of multiplications per sample needed by Algorithm 2 is more than 400 when block size is set to 1024 samples. To find out the optimal block size, we try to find the minimum value of the complexity equation of Algorithm 2.

From Section 2.1, we know that the number of complex multiplications per sample is

2FFT(2*N*)/*N*+*M*+*M*/*N*. From 錯誤! 找不到參照來源。, we know that for *N*-point real FFT, the number of complex multiplications needed is $(N/4)(\log_2 N + 3) - 1$. let *M* be approximated as *L*/*N*. The complexity equation is

$$\mathrm{C}(N) = \log_2 N + 4 + (L-2)N^{-1} + LN^{-2} \tag{48}$$

Differentiating C(*N*) with respect to *N* leads to

$$\mathrm{C}'(N) = \frac{1}{N \ln 2} - (L-2)N^{-2} - 2LN^{-3} \tag{49}$$

The optimum block length $N_{opt}$ can be obtaining through C'(*N*) = 0; that is

$$\frac{{N_{opt}}^2}{\ln 2} - (L-2)N_{opt} - 2L = 0 \tag{50}$$

Hence

$$N_{opt} = \left[ L - 2 + \sqrt{(L-2)^2 + \frac{8L}{\ln 2}} \right] \cdot \frac{\ln 2}{2} \tag{51}$$

In other words, the block length with best computation efficiency can be obtained if the filter length or the reverberation length is known. For example, when $L = 88200$, $N_{opt} \approx 61140$. Since *N* should be limited to be the power of two and the most often reverberation length is in the range 2-3 seconds. Another important issue is the length of the filter is directly proportional the block length. Furthermore, from *Figure 2.3*, the reduction complexity difference for *N* above 4000 is less than 10%. The block length considering all the above tradeoff is 4096.

## 2.3 Latency

Since the FFT needs to collect a segment to process for an output segment, the FFT-based convolution system have a latency with the same length of the FFT. In some applications like karaoke, the latency of reverberation may not be allowed. To solve this problem, FFT-based convolution methods can be modified by combining with direct convolution to remove the latency.

Consider on Algorithm 2, to shorten the latency, we use direct convolution to calculate the output segment of first impulse response segment. From (17), the output segment $y_p[n]$ can be expressed as

$$\begin{aligned} y_p[n] &= \sum_{k=0}^{N-1} x[n+pN-k]h[k] \\ &\quad + \sum_{s=1}^{M-1} x_{p-s-1}[n+N] * h_s[n] + \sum_{s=1}^{M-1} x_{p-s}[n] * h_s[n] \end{aligned} \tag{52}$$

For first sample of $y_p[n]$, $y_p[0] = y[pN]$, the inputs of the computation are $x_k[n]$, $p-1 \geqq k \geqq p-M+1$ and $x[n]$, $pN \geqq n \geqq pN-N+1$. The computation of $\sum_{s=1}^{M-1} x_{p-s-1}[n+N] * h_s[n]$ is completed while computing $y_{p-1}[n]$ if using overlap-and-add method. Since we already have these inputs when we get $x[pN]$, we can calculate $y_p[0]$ without waiting any other input samples. So are other samples in $y_p[n]$.

Though the implementation of (52) can remove the latency, the computation of $x_{p-1}[n]*h_1[n]$ can only be calculated after we get the sample $x[pN-1]$, the last sample of $x_{p-1}[n]$. If we want the application to be without any latency, the computation needed to be completed in a sampling period. This causes the demand on the processor to become non-uniform over time. To make the demand on the processor to be uniform, we can make use the direct convolution to calculate the output of first two segments of impulse response. Thus (52) can be expressed as

$$\begin{aligned} y_p[n] &= \sum_{k=0}^{2N-1} x[n+pN-k]h[k] \\ &\quad + \sum_{s=2}^{M-1} x_{p-s-1}[n+N] * h_s[n] + \sum_{s=2}^{M-1} x_{p-s}[n] * h_s[n] \end{aligned} \tag{53}$$

After this modification, the computation of FFT convolution can be finished in an input segment of time, just like the original algorithm.

Know that a direct convolution of *N*-point impulse response needs *N* multiplications for each output sample. Thus after this modification, the computational power requirement increases. For example, using Algorithm 2 with 4,096 block size for 88,200 samples of impulse response, it originally needs about 100 multiplications to compute an output sample. After this modification, it may need more than 8,000 multiplications to calculate an output sample.

## 2.4 Perceptual Convolution

The threshold in quiet is the threshold to characterize the minimum amount of energy needed in pure tone detected by human hearing system in a noiseless environment. *Figure 2.4* shows the threshold by Painter and Spanias [3].

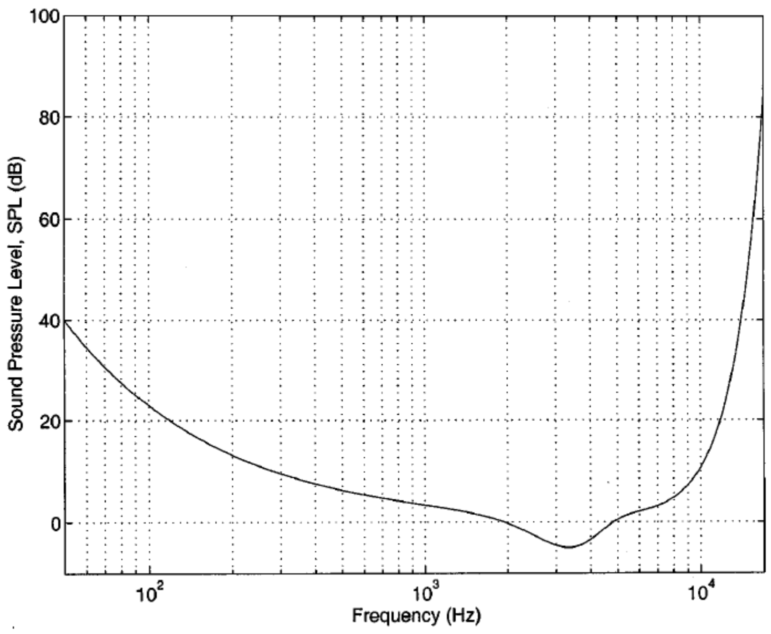

*Figure 2.4: The threshold in quiet (by Painter and Spanias [3])*

If we do the frequency analysis on an impulse response of a natural environment as shown in *Figure 2.5*, we can see that the higher frequency part will decay faster than lower frequency part. After partitioning the impulse response, the magnitude of higher frequency part of later blocks will be very small. In FFT convolution, the multiplications for those frequencies that its magnitude is smaller than 0dB can be removed, since the result will be ignored after IFFT. According to perceptual threshold, not only the multiplications for those frequencies can be removed, but also the multiplications for those frequencies in the higher frequency part that their magnitudes do not exceed the threshold can be removed.

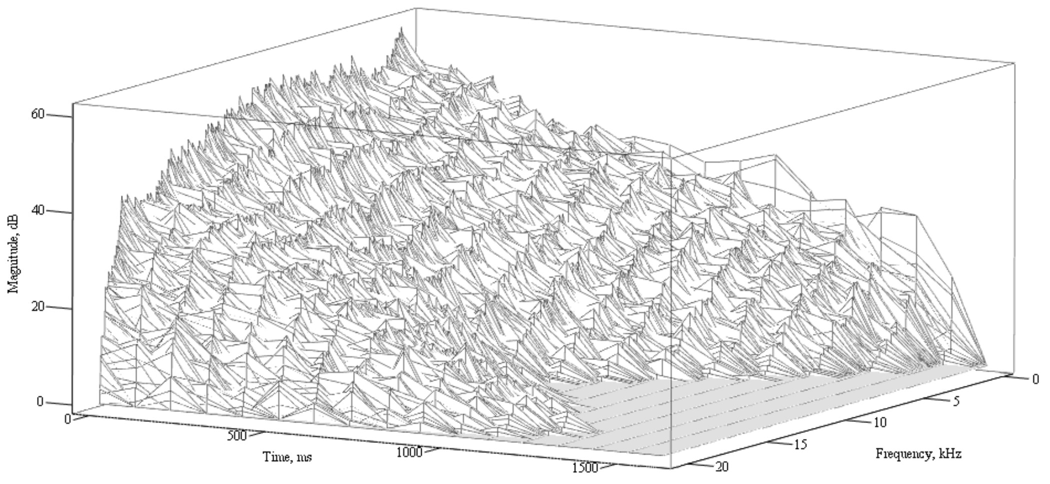

*Figure 2.5: Spectrum of the impulse response recorded from St. John Lutheran Church*

To implement the fast perceptual convolution, we need to decide the frequency part that can be removed. In Step 1 of Algorithm 1 or 2, we can get the frequency domain data of each small block in the impulse response. For each small block, we can calculate the magnitude of each frequency sample. Then, we scan from the highest frequency to find a frequency point in which its magnitude is equal or bigger than the perceptual threshold. In Step 3 of both algorithms, we can ignore the multiplications for those frequencies that are higher than the frequency point corresponding to each block found in Step 1.

*Table 2.1* shows the cutoff frequency point found in each block of 4 different impulse responses. For those impulse responses, we can eliminate more than 50% of multiplications in frequency domain. For some blocks, we can remove the multiplications for the whole block. *Figure 2.6* shows the same impulse response as that in *Figure 2.5* after removing ignored frequencies.

*Table 2.1: Cutoff frequency point of each block of each impulse response when the block size is set to 4,096*

| Impulse Response | | St John Lutheran 40 | Foellinger Great Hall 80 | Bethel Church 50 | Meyerson Concert Hall |
|---|---|---|---|---|---|
| Block count | | 17 | 17 | 17 | 15 |
| Block Index | 1 | 3052 | 3000 | 2992 | 2996 |
| | 2 | 2956 | 2964 | 2976 | 2892 |
| | 3 | 2896 | 2904 | 2900 | 2784 |
| | 4 | 2812 | 2812 | 2828 | 2632 |
| | 5 | 2716 | 2708 | 2680 | 1708 |
| | 6 | 2512 | 2544 | 1692 | 1548 |
| | 7 | 1652 | 1644 | 1476 | 1312 |
| | 8 | 1468 | 1544 | 1332 | 1244 |
| | 9 | 1264 | 1336 | 1244 | 1104 |
| | 10 | 1156 | 1220 | 1140 | 1088 |
| | 11 | 1088 | 1136 | 1024 | 996 |
| | 12 | 976 | 1104 | 992 | 920 |
| | 13 | 928 | 1048 | 956 | 876 |
| | 14 | 788 | 948 | 856 | 760 |
| | 15 | 568 | 860 | 800 | 0 |
| | 16 | 0 | 544 | 732 | |
| | 17 | 0 | 0 | 544 | |
| Eliminated Percentage | | 61.47% | 59.33% | 60.99% | 62.79% |

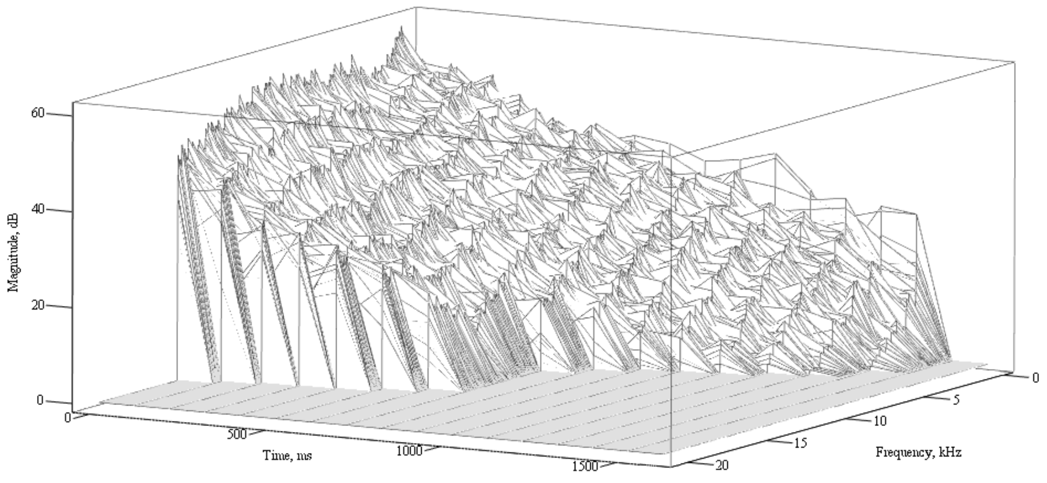

*Figure 2.6: Spectrum of the impulse response of St. John Lutheran Church after applying the perceptual threshold*

## 2.5 Low-Delay Reverberators

Since the FFT needs to accumulate a segment to begin the FFT computation, the FFT-based convolution introduced an additional algorithm delay or latency by one FFT block, that is *N*. In some real-time applications like interactive environment, the latency should be restricted. In the literature, there have been developed methods, such as [4][20], to shorten the latency of the filter by using time domain filter said low latency filter to compute the output of the first impulse response segment.

To remove the latency of the FFT-based convolution filters, they can be modified by combining with direct convolution to remove the latency. According to [4] the length of the time-domain filter should be twice of the block size of the FFT-based convolution filter to make its demand on the processor to be uniform over time. With block size of 4096, we need to have extra 8192 multiplications to make the application to be zero-delay.

To reduce the complexity, there are two methods can be used. The first method is to use smaller block size of FFT-based convolution filter. The second is adding another FFT-based convolution filter with smaller block size. For the first method, the optimal block size is 512 when the filter length is set to 88200, and it needs about 1760 multiplications per sample. For the second method, the optimal block size of the smaller filter is 128, and it needs only about 700 multiplications per sample.

## 2.6 Real-Time Reverberators Analysis and Implementation

Assuming that we can remove 60% of multiplications in frequency domain, we can calculate the number of multiplications needed for fast perceptual convolution by modifying the complexity from Algorithm 2 as illustrated in *Figure 2.7*. From the result, the fast perceptual convolution requires about 98 real multiplications per sample to convolve with 88,200 samples of impulse response.

To evaluate the improvement in real-time systems, we built an experiment application to help us finishing the test. The application will use two methods, the fast perceptual convolution method and Algorithm 2, to process some samples for comparison. The input block size is set to 4,096. And the test is to process single channel, 4,096×20,000 = 81,920,000 samples of input, which is about 30 minutes of samples with 44,100Hz sampling rate. The test is run on a PC with 1GHz Pentium*!!!*. The result is listed in *Table 2.2*.

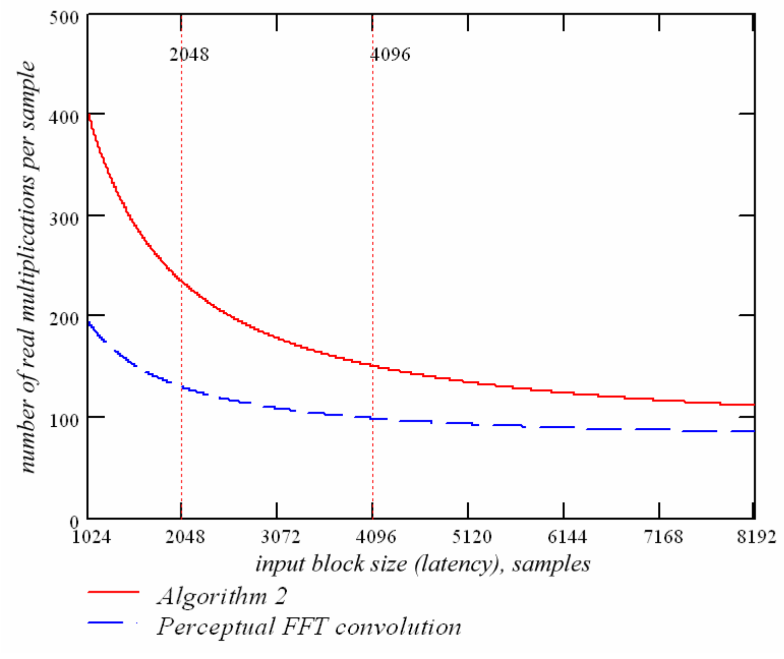

*Figure 2.7: Comparison of fast perceptual convolution and Algorithm 2 when the length impulse response is 2 seconds (88,200 samples)*

*Table 2.2: Comparison of fast perceptual convolution and Algorithm 2*

| Time in ms | St John Lutheran 40 | Foellinger Great Hall 80 | Bethel Church 50 | Meyerson Concert Hall |
|---|---|---|---|---|
| Algorithm 2 | 89469 | 88027 | 84692 | 82549 |
| Fast perceptual convolution | 59566 | 61057 | 58694 | 57032 |
| Improved Ratio | 33.42% | 30.64% | 30.70% | 30.91% |

*Table 2.2* shows that the fast perceptual convolution can reduce about 30% complexity as compared with the Algorithm 2 in real applications.

## 2.7 Objective and Subjective Measurement

The fast perceptual convolution exploits the perceptual irrelevancy to develop fast convolution. This chapter considers the objective measure to check the irrelevancy. The Objective Difference Grade (ODG) which is suggested by Recommendation ITU-R BS.1387 [5] is introduced to the measurement.

## 2.8 Objective Difference Grade

The ODG is the output variable from the objective measurement method and corresponds to the Subjective Difference Grade (SDG, Recommendation ITU-R BS.1116) in the subjective domain. The method is a perceptual measurement method for audio signal processing to determine the perceptual difference between the two input signals, i.e. the Reference Signal (RS) and the Signal Under Test (SUT). The value should ideally range from 0 to -4, where the value 0 corresponds to an imperceptible impairment and -4 to an impairment judged as very annoying. The result value is negative, because the SUT's quality is assumed to be worse than RS's. But in our experiment, we compare the result of fast perceptual convolution and the result of generic convolution methods. Because the perceptual quality of fast perceptual convolution will not always worse than generic convolution methods, the ODG will vary from −4 to 4.

## 2.9 Comparison with Generic Convolution Methods

To measure the reverberation quality of fast perceptual convolution, we use ODG to compare the reverberation generated by generic convolution methods and fast perceptual convolution. The result of is listed in *Table 2.3*.

*Table 2.3: ODG results of the fast perceptual convolution compared to generic convolution methods*

| | St John Lutheran 40 | Foellinger Great Hall 80 | Bethel Church 50 | Meyerson Concert Hall |
|---|---|---|---|---|
| | ODG | ODG | ODG | ODG |
| '69.wav | 0 | 0 | -0.01 | 0.01 |
| 1k+5k.wav | 0.03 | 0.02 | 0.03 | 0.04 |
| 60.wav | 0.03 | 0.04 | 0.05 | 0.03 |
| 9_1.wav | -0.02 | -0.03 | -0.03 | -0.02 |
| 9_2.wav | -0.02 | -0.03 | -0.03 | -0.02 |
| 9_3.wav | -0.02 | -0.03 | -0.03 | -0.02 |
| applaud.wav | 0 | -0.01 | -0.01 | -0.01 |
| A_DAY_FOR_YOU.wav | 0 | -0.01 | -0.01 | 0 |
| BlackBird.wav | 0 | 0 | 0 | 0 |
| butter1.wav | 0.03 | 0.03 | 0.03 | 0.03 |
| castanets.wav | -0.02 | -0.02 | -0.02 | -0.01 |
| cello1.wav | -0.01 | -0.01 | -0.01 | -0.01 |
| coco.wav | -0.01 | -0.01 | -0.01 | 0 |
| dance1.wav | 0.02 | 0.02 | 0.02 | 0.02 |
| else3.wav | -0.01 | -0.01 | -0.01 | -0.01 |
| fatboy.wav | 0.01 | 0.01 | 0.01 | 0.01 |
| flute.wav | -0.01 | -0.01 | -0.01 | -0.01 |
| Fools.wav | 0 | 0 | 0 | 0 |
| ftb_samp.wav | 0 | -0.01 | -0.01 | 0 |
| goldc.wav | 0 | -0.01 | -0.02 | 0 |
| gong.wav | -0.01 | -0.01 | -0.01 | -0.01 |

| | | | | |
|---|---|---|---|---|
| gong2.wav | -0.03 | -0.03 | -0.03 | -0.03 |
| harp.wav | -0.01 | -0.01 | -0.01 | -0.01 |
| hat1.wav | 0 | -0.01 | -0.01 | 0 |
| heart.wav | 0 | 0 | 0 | 0 |
| HEART1.wav | 0 | 0 | 0 | 0 |
| Hero.wav | 0.01 | 0 | 0 | 0 |
| Hero2.wav | 0 | 0 | 0 | 0 |
| hihat.wav | -0.01 | -0.01 | -0.07 | -0.1 |
| iron.wav | -0.01 | -0.01 | -0.01 | -0.01 |
| KMFDM-Dogma.wav | 0 | -0.01 | -0.01 | 0 |
| land.wav | -0.01 | -0.01 | -0.01 | -0.01 |
| leftright.wav | -0.01 | -0.01 | -0.01 | -0.01 |
| main_theme.wav | 0.03 | 0.03 | 0.03 | 0.03 |
| man.wav | 0.02 | 0.02 | 0.02 | 0.02 |
| memory.wav | 0.03 | 0.02 | 0.02 | 0.03 |
| mist.wav | 0.03 | 0.03 | 0.03 | 0.03 |
| Moonly.wav | -0.03 | -0.04 | -0.04 | -0.03 |
| mstest.wav | -0.01 | -0.01 | -0.01 | -0.01 |
| mvoice.wav | 0.04 | 0.04 | 0.04 | 0.04 |
| pipes.wav | -0.01 | -0.01 | -0.01 | -0.01 |
| point1.wav | -0.01 | -0.02 | -0.02 | -0.02 |
| spahm.wav | 0.02 | 0.02 | 0.02 | 0.02 |
| st_jacob.wav | 0 | 0 | -0.01 | 0 |
| summer.wav | -0.01 | -0.02 | -0.02 | -0.01 |
| t1.wav | 0 | 0 | 0 | 0 |
| testsignal2.wav | 0.03 | 0.03 | 0.03 | 0.03 |
| testsignal4.wav | -0.01 | -0.01 | -0.01 | -0.01 |
| The_red_Sorghum.wav | 0 | 0 | 0 | 0 |
| This_Land_(Instrumental)-short.wav | 0 | -0.01 | -0.01 | 0 |
| This_Land_(Instrumental).wav | 0 | -0.01 | -0.01 | 0 |
| tpd.wav | -0.01 | -0.02 | -0.02 | -0.01 |
| Track07.wav | -0.02 | -0.02 | -0.02 | -0.02 |
| track7.wav | -0.01 | -0.01 | -0.01 | -0.01 |
| tsai.wav | -0.01 | -0.01 | -0.01 | -0.01 |
| vbrtest.wav | -0.01 | -0.02 | -0.02 | -0.01 |
| velvet.wav | 0.01 | 0.01 | 0.03 | 0.01 |
| WINTER.wav | -0.01 | -0.02 | -0.02 | -0.01 |

| | | | | |
|---|---|---|---|---|
| wvoice.wav | 0.03 | 0.03 | 0.03 | 0.03 |
| youcantdothat.wav | 0 | -0.01 | -0.01 | 0 |
| **MEAN ABSOLUTE ODG** | 0.01217 | 0.01483 | 0.017 | 0.01383 |
| **STANDARD DEVIATION** | 0.0119071 | 0.0109583 | 0.011424 | 0.0112997 |

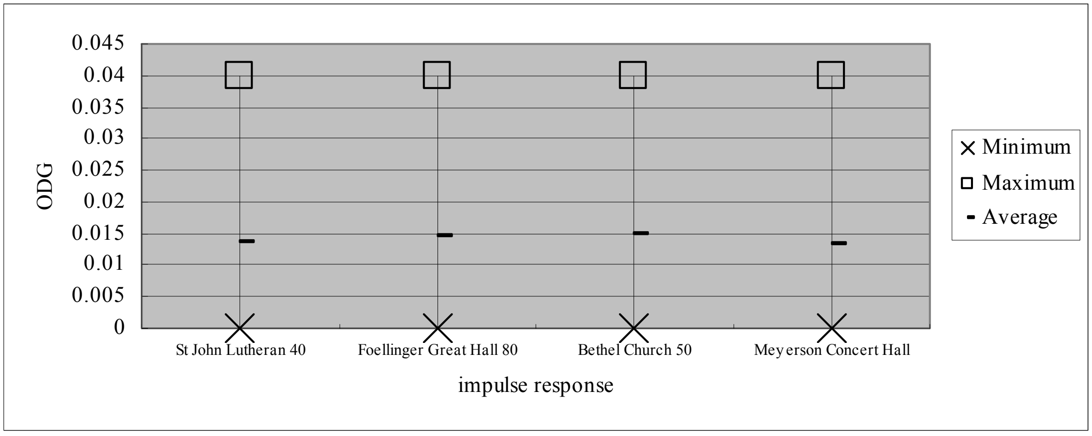

*Figure 2.8: Distribution of ODGs for fast perceptual convolution with different impulse responses*

As shown in *Table 2.3* and *Figure 2.8*, the mean absolute ODG for each impulse response are smaller than 0.015 and the maximum ODG are all 0.04. These results show that the differences between the outputs of the proposed method and the original method are not perceptually noticeable. The result shows that the fast perceptual convolution has a speedup 30% over the FFT-based convolution without scarifying the reverberation quality.

## 2.10 Measurement of Different Levels of Thresholds

To reduce more the multiplications, we can use higher level of perceptual threshold curve. In the following, we will measure the ODG results for the fast perceptual convolution with different levels of perceptual threshold. In this test, we use the impulse response of St. John Lutheran Church. The result is listed in *Table 2.4*.

*Table 2.4: ODG results of the fast perceptual convolution using different levels of perceptual threshold*

| level (dB) | 0 | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Reduced Ratio | 61.47% | 63.34% | 65.98% | 67.58% | 70.60% | 72.13% | 74.66% | 77.74% | 79.74% | 83.11% | 85.45% |
| '69.wav | 0 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 | -0.09 | -0.07 | -0.11 | -0.16 |
| 1k+5k.wav | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.02 | 0.02 | 0.01 |
| 60.wav | 0.03 | 0.04 | 0.04 | 0.04 | 0.04 | 0.05 | 0.04 | 0.04 | 0.05 | 0.05 | 0.02 |
| 9_1.wav | -0.02 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | 0 | 0.04 | 0 | 0.1 |
| 9_2.wav | -0.02 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | 0 | 0 | 0.01 | 0.07 |
| 9_3.wav | -0.02 | -0.01 | -0.01 | -0.01 | -0.17 | -0.16 | 0 | 0.02 | 0.06 | 0.07 | 0.06 |
| applaud.wav | 0 | 0 | 0 | 0 | 0 | 0 | 0.05 | 0.06 | 0.03 | 0.02 | 0.01 |
| A_DAY_FOR_YOU.wav | 0 | 0 | 0 | 0.01 | -0.17 | -0.06 | -0.02 | -0.01 | 0 | -0.08 | -0.13 |
| BlackBird.wav | 0 | 0 | 0 | 0 | -0.02 | -0.07 | -0.11 | -0.47 | -0.49 | -0.49 | -0.58 |
| butter1.wav | 0.03 | 0.03 | 0.03 | 0.03 | 0.04 | 0.04 | 0.04 | 0.04 | 0.06 | 0.05 | 0.05 |
| castanets.wav | -0.02 | -0.01 | 0 | -0.1 | -0.08 | -0.08 | -0.11 | -0.16 | -0.17 | -0.28 | -0.28 |
| cello1.wav | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | 0 | 0.01 | 0.08 | 0.06 | 0.08 |
| coco.wav | -0.01 | 0 | 0 | 0 | 0 | 0 | 0.06 | 0.07 | 0.06 | 0.08 | 0.08 |
| dance1.wav | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.04 | -0.06 | 0.04 | 0.03 | -0.01 |
| else3.wav | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | 0 | -0.13 | -0.12 | -0.1 | -0.11 |
| fatboy.wav | 0.01 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | -0.01 | 0 | 0.02 | -0.07 | -0.19 |
| flute.wav | -0.01 | -0.01 | -0.01 | -0.01 | -0.09 | -0.09 | 0 | -0.01 | 0.03 | 0.04 | 0.03 |
| Fools.wav | 0 | 0.01 | 0.01 | -0.06 | 0 | 0 | -0.04 | 0.01 | 0 | -0.03 | -0.09 |
| ftb_samp.wav | 0 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | -0.02 | -0.03 | -0.02 | -0.04 | -0.03 |
| goldc.wav | 0 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 |
| gong.wav | -0.01 | -0.01 | -0.01 | 0 | -0.01 | 0 | -0.01 | -0.06 | -0.04 | -0.05 | 0.01 |
| gong2.wav | -0.03 | -0.03 | -0.03 | -0.03 | -0.02 | -0.03 | -0.02 | -0.01 | 0.04 | 0.05 | 0.08 |
| harp.wav | -0.01 | -0.01 | -0.01 | -0.01 | 0 | 0 | 0 | -0.01 | 0.02 | 0.02 | 0.05 |
| hat1.wav | 0 | 0 | 0 | 0 | 0 | 0 | 0.01 | 0.01 | 0.02 | 0.1 | 0.1 |
| heart.wav | 0 | 0 | 0 | 0 | 0.01 | 0.01 | 0.06 | 0.05 | 0.07 | 0.07 | 0.05 |
| HEART1.wav | 0 | 0 | 0 | 0 | 0.01 | 0.01 | 0.06 | 0.05 | 0.07 | 0.07 | 0.05 |
| Hero.wav | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 | -0.09 | -0.07 | 0.02 | 0.03 | 0.05 | 0.08 |
| Hero2.wav | 0 | 0 | 0 | 0 | 0 | 0 | 0.01 | 0.02 | 0.06 | 0.08 | 0.1 |
| hihat.wav | -0.01 | -0.03 | -0.13 | -0.45 | -0.3 | -0.33 | -0.39 | -0.71 | -0.81 | -1.35 | -1.68 |
| iron.wav | -0.01 | 0 | -0.01 | -0.02 | -0.04 | -0.32 | -0.22 | -0.24 | -0.26 | -0.38 | -0.39 |
| KMFDM-Dogma.wav | 0 | 0.01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.01 | -0.08 |
| land.wav | -0.01 | -0.01 | -0.01 | -0.01 | 0 | 0 | 0 | 0.01 | 0.02 | 0.05 | 0.05 |
| leftright.wav | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.06 | -0.04 | -0.04 | 0.02 |
| main_theme.wav | 0.03 | 0.04 | 0.03 | 0.03 | 0.04 | 0.04 | 0.04 | 0.04 | 0.08 | 0.06 | 0.04 |
| man.wav | 0.02 | -0.02 | -0.02 | -0.02 | 0.03 | 0.04 | 0.04 | 0.05 | 0.02 | -0.09 | -0.18 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| memory.wav | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.04 | 0.04 | 0.04 |
| mist.wav | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.06 | 0.06 | 0.06 | 0.06 | 0.05 |
| Moonly.wav | -0.03 | -0.02 | -0.02 | -0.02 | 0 | -0.04 | -0.01 | 0.01 | 0.03 | -0.12 | -0.17 |
| mstest.wav | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 |
| mvoice.wav | 0.04 | 0.04 | 0.04 | 0.04 | 0.07 | 0.07 | 0.07 | 0.05 | 0.05 | 0.03 | -0.04 |
| pipes.wav | -0.01 | 0 | 0 | 0 | 0 | -0.01 | 0 | -0.01 | 0 | 0.01 | 0.05 |
| point1.wav | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | 0.02 | 0.03 | 0.03 | 0.07 | 0.06 |
| spahm.wav | 0.02 | 0.03 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.03 | -0.03 | -0.06 |
| st_jacob.wav | 0 | 0.01 | 0.01 | 0.01 | 0.01 | 0 | 0.01 | 0.01 | 0.01 | 0.02 | 0.06 |
| summer.wav | -0.01 | 0 | 0 | 0 | 0.01 | 0 | 0.01 | 0.02 | -0.13 | 0 | 0.04 |
| t1.wav | 0 | 0 | 0 | 0 | -0.02 | -0.04 | -0.05 | -0.51 | -0.56 | -0.35 | -0.42 |
| testsignal2.wav | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.04 | 0.03 | 0.04 | -0.09 | -0.18 |
| testsignal4.wav | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | 0.04 | 0.04 |
| The_red_Sorghum.wav | 0 | 0 | 0 | 0 | 0 | 0 | 0.01 | 0.01 | 0.05 | 0.06 | 0.07 |
| This_Land_short.wav | 0 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.05 |
| This_Land_.wav | 0 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 | 0.02 | 0.07 |
| tpd.wav | -0.01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.01 | 0.06 |
| Track07.wav | -0.02 | -0.02 | -0.02 | -0.02 | -0.01 | -0.02 | -0.01 | -0.19 | -0.13 | -0.12 | -0.13 |
| track7.wav | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | 0 | -0.07 | 0.01 |
| tsai.wav | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | 0 | 0.02 | 0.01 | 0 | 0.03 |
| vbrtest.wav | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | -0.01 | 0 | -0.09 | -0.08 |
| velvet.wav | 0.01 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0 | -0.04 | -0.08 | -0.2 | -0.3 |
| WINTER.wav | -0.01 | 0 | 0 | 0 | 0 | 0 | -0.02 | 0.02 | 0.03 | 0.02 | 0.02 |
| wvoice.wav | 0.03 | 0.04 | 0.04 | 0.04 | 0.07 | 0.07 | 0.06 | 0.08 | 0.07 | 0.05 | 0 |
| youcantdothat.wav | 0 | 0 | 0 | 0 | 0 | 0 | 0.01 | 0.06 | 0.04 | 0.08 | 0.07 |
| **MEAN ABSOLUTE ODG** | 0.0122 | 0.0132 | 0.0143 | 0.0223 | 0.0275 | 0.034 | 0.0342 | 0.0642 | 0.0732 | 0.0952 | 0.1197 |
| **STANDARD DEVIATION** | 0.011 | 0.01214 | 0.0191 | 0.0587 | 0.0493 | 0.0621 | 0.059 | 0.1263 | 0.1378 | 0.1883 | 0.2309 |

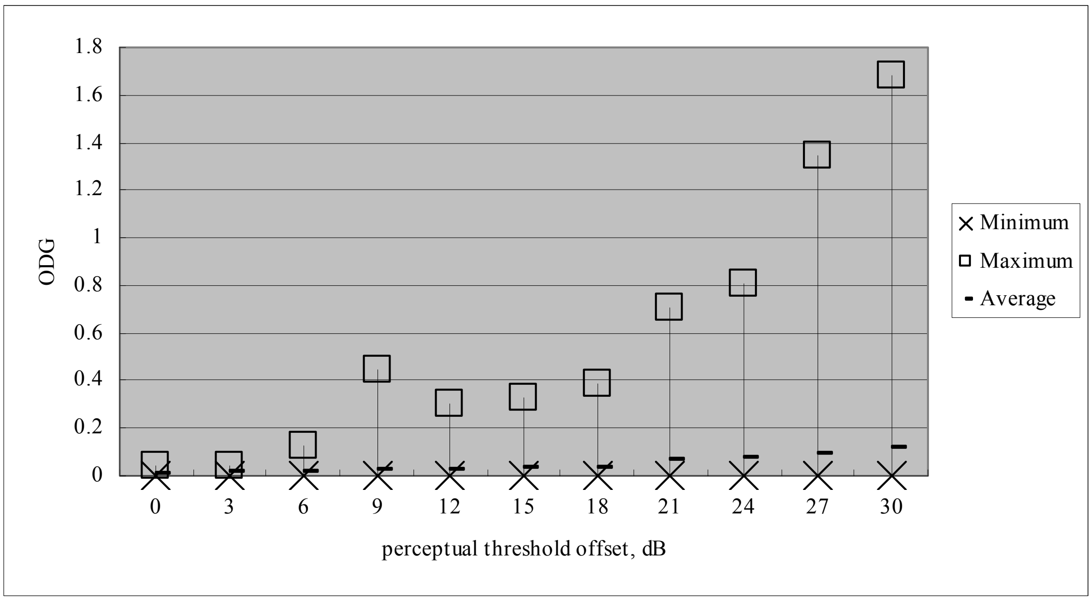

*Figure 2.9: Distribution of ODGs for fast perceptual convolution with different levels of perceptual threshold*

In *Table 2.4*, after we offset the perceptual threshold 30dB, we can remove about 80% of multiplications in frequency domain. By offsetting 30dB, the fast perceptual convolution can reduce 40% of complexity in multiplications as compared with the FFT-based convolution method. The mean absolute ODG is about 0.1. This result is still in acceptable range. The maximum ODGs are larger for few samples, especially for *hihat.wav*, because the overall frequencies of these samples are higher. Offsetting the threshold may filter out the important frequency components of those samples. By the same level of threshold, the fast perceptual convolution has a speedup of 66% over the FFT-based convolution when the input block size is set to be 1024.

# 3　自評完成之工作項目及成果

## 3.1　成果有以下四點

本計畫發表兩篇國際會議論文[8][13]，也已有一項專利正在申請當。本計畫對提出研究成果

（一）Formula Derivation of the Block-based FFT reverberators based on Overlap-and-add and Overlap-and-Save Methods，

（二）Fast Perceptual Reverberation Method，

（三）Low-Delay Perceptual Reverberation Method，

（四）Real-Time System Design for the IIR-based Reverberation and FIR-based Reverberation，

（五）Objective Measurement and Subjective Measurement System，

與原計畫相合

## 3.2 具體結果：

(1) 由於音樂廳對未來音效產業界有相當影響，因此，本計畫對累積此方面技術有其未來市場潛力及前瞻性。

(2) 本計畫有許多子題都有近年來學術研究焦點，相信學術上是值得研究的。

(3) 參與本計畫對人員，可獲得音訊處理經驗，對人才培育及技術累積有許多助益。

# References

[1] D. S. McGrath, "Method and Apparatus for Filtering an Electronic Environment with Improved Accuracy and Efficiency and Short Flow-Through Delay", US Patent 5,502,747.

[2] J. H. McClellan, Ronald W. Schafer & Mark A. Yoder, "DSP First: A Multimedia Approach", Prentice Hall, 1998, ISBN 0-13-243171-8

[3] T. Painter and A. Spanias, "Perceptual Coding of Digital Audio", Proceeding of The IEEE, Vol. 88. No. 4, April 2000.

[4] W. G. Gardner, "Efficient Convolution without Input-Output Delay", J. Audio Eng. Soc., vol. 43, no. 3, pp. 127-136, March 1995.

[5] ITU Radiocommunication Study Group 6, "DRAFT REVISION TO RECOMMENDDATION ITU-R BS.1387 - Method for objective measurements of perceived audio quality".

[6] M. R. Schroeder and B. F. Logan, "Colorless Artificial Reverberation", J. Audio Eng. Soc., vol. 9, no. 3, pp. 192-197, July 1961.

[7] M. R. Schroeder, "Natural Sounding Artificial Reverberation", J. Audio Eng. Soc., vol. 10, no. 3, pp. 219-223, July 1962.

[8] J. A. Moorer, "About This Reverberation Business", Computer Music Journal, vol. 3, no. 2, pp. 13-28, June 1979.

[9] J. M. Jot and A. Chaigne, "Digital delay networks for designing artificial reverberators", in Proc. 90th Conv. Udio Eng. Soc., February, 1991, preprint 3030.

[10] W. G. Gardner, "The virtual acoustic room", MS proposal, MIT Media Lab, 1992.

http://alindsay.www.media.mit.edu/projects.html.

[11]J. Dattorro, “Effect Design Part 1: Reverberator and Other Filters”, J. Audio Eng. Soc., vol. 45, pp. 660-684, September 1997.

[12] H. V. Sorensen, D. L. Jones, M. T. Heideman, and C. S. Burrus, “Real-Valued Fast Fourier Transform Algorithms”, IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-35, June 1987.

[13] W. C. Sabine, “Reverberation”, in Lindsay, R.B., editor, Acoustic: Historical and Philosophical Development, Stroudsburg, PA: Dowden Hutchinson, and Ross, 1972. Originally published in 1990.

[14] T. G. Stockham, Jr., “High-Speed Convolution and Correlation”, in Spring Joint Computer Conf., AFIPS Conf. Proc., vol. 28, pp. 229-233, 1966; reprinted in Digital Signal Processing, Selected Reprints, L. R. Rabiner and C. M. Rader, Eds. (IEEE Press, New York, 1972).

[15] Sean Browne, “Hybrid Reverberation Algorithm Using Truncated Impulse Response Convolution and Recursive Filtering”, MS proposal, Music Engineering Technology, University of Miami, June 2001.

[16] A. V. Oppenheim and R. W. Schafer, “Digital Signal Processing”, Prentice-Hall, Englewood Cliffs, NJ, 1975

[17] K. Iida, K. Mizushima, Y. Takagi, and T. Suguta, “A New Method of Generating Artificial Reverberant Sound”, 99th AES Convention 1995 October 6-9, 1995.

[18] Anders Torger and Angelo Farina, “Real-Time Partitioned Convolution for Ambiophonics Surround Sound”, IEEE Workshop on Applications of Signal Processing to Audio an Acoustics 2001, October 2001.

[19] J. S. Soo, K. K. Pang, “Multidelay block frequency adaptive filter”, IEEE Trans. Acoust. Speech Signal Process., Vol. ASSP-38, No. 2, February, 1990.

[20] Electronic Environment with Improved Accuracy and Efficiency and Short Flow-Through Delay”, US Patent 5,502,747