整合 MIDI 伴奏之國語歌聲合成系統

A Mandarin Singing Voice Synthesis System with MIDI-Accompanying Integrated

古鴻炎　　陳安璿　　廖皇量
Hung-Yan Gu　　*An-Siuen Chen*　　*Huang-Liang Liao*

國立台灣科技大學資訊工程系
e-mail: guhy@mail.ntust.edu.tw　　http://guhy.csie.ntust.edu.tw

摘要

在本研究中，我們以加式弦波模型為基礎，作模型的擴充與改進，以合成國語音節的歌聲信號，及合成音符間的轉音和音符內的抖音的效果。依此歌聲合成模型，我們製作了一個整合 MIDI 伴奏之國語歌聲合成系統，它可依 MIDI 檔裡的主旋律音符和歌詞訊息，來合成出國語歌聲，再和 MIDI 伴奏音符作同步、即時之演奏。在以聽測實驗作評估後，結果顯示我們的模型所合成出的歌聲配上 MIDI 伴奏，在自然度、清晰度、及喜好度方面，都比未加上伴奏時的同一模型或 TIPW 法好很多。

1. 前言

由於個人電腦的普及，若能在電腦上執行歌聲合成軟體，人們將可方便地學習新的歌曲、老歌、或民謠。此外作詞、作曲者，也可藉由歌聲合成軟體，來快速地聆聽、評估自己的作品。更進一步，若能將 MIDI 樂器演奏功能及歌聲合成功能結合在一起，來進行同步、即時的演奏，則可達到有如電腦在唱卡拉 OK 一樣的效果，讓電腦變得更爲人性化，並且有 MIDI 伴奏的電腦合成歌聲，聽起來也會比較悅耳，就如人們唱歌時需要伴奏一樣。

在歌唱聲合成方面，國外已有一些使用當地語言的歌聲合成之研究成果可供參考[1, 2, 3, 4, 5]，不過由於語言的差異，國外的研究成果，並不能夠直接就套用於國語上。在國內，也有一些國語歌聲合成的研究成果被提出[6, 7, 8, 9]，所以國語歌聲合成是有不少人在耕耘的領域，不過尚未有人嘗試製做整合 MIDI 樂器演奏及國語歌聲合成的同步、即時演奏系統。

本研究的目標是：(a)研究國語歌聲合成之技術；(b)製作具有 MIDI 伴奏之整合式的國語歌聲合成系統，能以同步、即時的方式來演奏。關於國語歌聲的合成，我們以加法式合成(additive synthesis)之弦波模型爲基礎[10, 11]，來考慮和解決如下的幾個問題：(1)音色(timbre)一致性，不管音高(基本頻率)變高或變低，音色都需保持爲同一個人的[10]；(2)轉音之唱法，允許一個字唱 2 個甚至 3 個音符，這比起一個字唱一個音符是較難處理的；(3)時長(duration)訂定和無聲子音合成，無聲子音(如/s, ts/)若處理不當，不僅子音本身變得不清楚，也會使拍子、節奏錯掉；(4)抖音(vibrato)模擬，抖音是真人歌聲裡會聽到的現象，發生抖音的主要原因是，音高變得不固定而隨著時間作慢速擺動[4, 10, 11]。

國語歌聲和 MIDI 樂器之兩種演奏要作整合，意味我們需要以 MIDI 檔裡記錄的旋律及歌詞資訊，來合成出各個歌詞字的歌聲信號，這牽涉到兩個問題：(1)歌詞字如何與主旋律音符對應；(2)歌詞字若是破音字，要如何決定其發音音節。此外，若要讓兩者進行同步的演奏，則也需考慮使用那一種機制來達到同步。至於即時性，一般來說只要電腦速度夠快，應不成問題。

在本研究裡，我們主要探討了前述的幾個問題，然後依據所提出的解決方法，來製做一個整合的系統，以作驗證。我們的系統，其主處理流程如圖 1 所示，”MIDI 檔輸入” 方塊讀入一個 MIDI 檔的資料後，”MIDI 檔分析”方塊會先詢問使用者主旋律所在的聲道(channel)，然後把主旋律音符、歌詞、和其它的 MIDI 伴奏音符分離出來，並且存到各自的資料串列(list)；接者，下一個方塊作歌詞與音符配對、歌詞轉音節的處理，之後，起動一個新的執行序(thread)，去作歌聲信號合成的工作，如圖 1 中向右的虛線表示新執行序；至於向左的實線，則是原先的執行序，繼續去作同步播放的處理。關於歌聲信號合成的方法，將在第 2 節裡說明，而歌詞與音符配對、歌詞轉音節，將在第 3 節裡說明，同步播放之處理，則在第 4 節中說明。

2. 歌聲合成方法

依據我們過去的研究經驗[8, 12, 13]，不管使用時域或頻域上的方法，所合成出的國語歌聲都仍然有缺

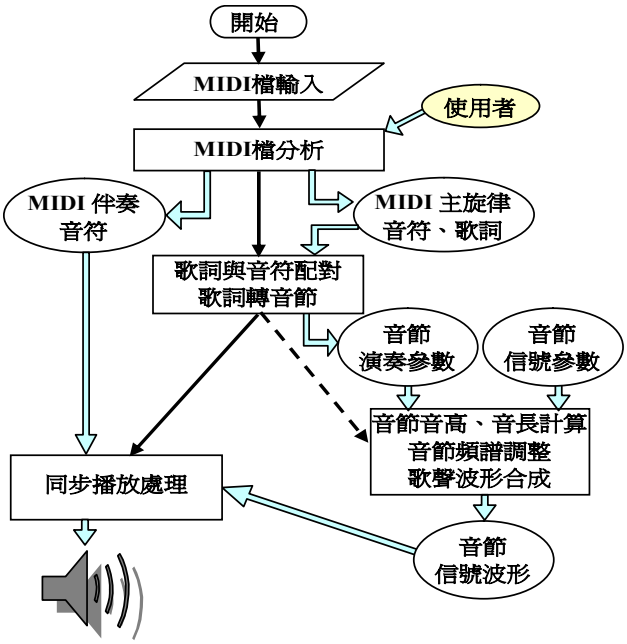

圖 1 本系統的主處理流程

點，例如時域方法所合成出的聲音，會比較呆滯而不夠靈活，而弦波模型方法合成出的聲音，則會有類似金屬聲音的感覺，且當音高變得較高或較低時，音色也會產生一些改變。不過兩者比較起來，加法式弦波模型之合成方法，比較能夠合成出活靈活現的歌唱聲，因此在本研究裡，我們還是選擇以加法式弦波模型的方法作爲基礎，再據以作改進，以加入更多的功能。

2.1. 弦波模型

所謂加法式弦波模型，其一般化的公式爲：

$$s[n]=\sum_{h=0}^{H-1} A_h[n]\cdot cos\left(2\pi\cdot f_h\cdot\frac{n}{F_s}+\theta_h\right) \qquad (1)$$

其中 $s[n]$表示第 n 個樣本時刻上的信號樣本值，h 爲諧波編號，$A_h[n]$ 表示第 h 個諧波在樣本時刻 n 時的振幅(即爲時變的)，f_h 表示第 h 個諧波的頻率值，而 θ_h 則是第 h 個諧波的初始相位，F_s 是取樣頻率。

考慮到轉音與抖音的合成處理，實作上公式(1)並不適合直接用來計算信號樣本的數值，因此我們首先把公式(1)改換成如下之形式[10, 11]

$$s[n]=\sum_{h=0}^{H-1} A_h[n]\cdot cos\left(\omega_h[n]+\theta_h^0\right) \qquad (2)$$

其中 $\omega_h[n]$表示第 h 個諧波從樣本時刻 0 到時刻 n 所累積起來的相位量，θ_h^0 表示從第一個音框分析得到的初始相位。早先在電腦音樂合成的研究領域，以公式(2)取代公式(1)，是爲了把乘算換成加算以節省計算量。接者依據 $\omega_h[n]$ 的定義，可知 $\omega_h[n]$ 會滿足如下之遞迴關係：

$$\omega_h[n] = \omega_h[n-1] + \Delta\omega_h[n] \qquad (3)$$

其中 $\Delta\omega_h[n]$ 表示第 h 個諧波在樣本時刻 n 時的相位增量，如果先不考慮轉音、抖音效果的合成，則 $\Delta\omega_h[n]$ 可簡化成 $\Delta\omega_h$ ，即和時刻 n 無關，而它的數值可由下列公式求得：

$$\Delta\omega_h = (2\pi\cdot f_h)/F_s \qquad (4)$$

即由諧波頻率 f_h 來決定。

2.2. 時長調整

2.2.1. 起始時間點調整

對於以子音開頭的的音節來說，把音符的起始時間作爲音節的起始時間，所合成出的音節信號，聽起來拍子與配樂音符不是同步的，即聽起來會有拖拍的感覺。關於拖拍的問題，我們的解決方法是，建立一個子音之時間平移表，記錄各種子音的平移時間，再把音節按照對應子音的平移時間，向前移動起始時間點，並且修改時間長度。

2.2.2. 換氣模擬

當相鄰音符之間沒有停頓，而我們又直接依據音符的音長來作音節信號合成，會使歌詞聽起來就像是整串連在一起，缺少人類歌唱時自然換氣、停頓的感覺。我們模擬換氣的方法是，將每一個音符的音長均乘以一個長度調整參數β，當音符長度超過 1.3 秒，就將β值設爲 0.94，否則設爲 0.85。

2.2.3. 子音長度調整

在我們的系統中，無聲子音的信號波形並未以弦波模型來產生，而是在時域上對原信號波形直接作長度調整、及串接的處理。當合成的子音太短，則一種基本的子音伸長或縮短的方法[14]，會造成子音強度變弱，而使得聽起來變不清楚。因此我們研究了另一種作法，不管伸長或縮短子音，都先將原始與合成子音分割成音框序列，以音框爲單位去截取原始子音中對應的音框，取回後作相鄰音框間的疊加。如此就可以保留原始子音波形的振幅包絡，而使合成的子音仍能一樣清楚[15]。

2.2. 音高、諧波振幅調整

要讓音高(基本頻率)升高或降低，不能夠採取 resampling 作法[16]，因爲會造成音色隨著音調高低在改變。音高是由第一個諧波的頻率值(基本頻率)或相鄰諧波間的頻率差距所決定，而音色則是由各諧波的振幅峰點所連接成的包絡(envelop)曲線的形狀所決定

[10]。因此在每一種音節只錄一遍發音的情形下，當要調整某一個諧波的頻率位置時，我們也必須對其振幅高度作調整，以保持包絡曲線爲固定不變的形狀，如此音色就可維持不變。

當要決定一個新的頻率上的諧波的振幅，則需要依據先前記錄的諧波參數來作內插，在此使用三階之 Lagrange 內插方法[17]，其圖形說明如圖 2 所示，其中 x 表示一個新的諧波頻率，y 表示 x 上內差出的振幅值，x_0, x_1, x_2, x_3 表示 x 兩邊最接近 x 的 4 個原音節中的諧波頻率值，y_j 則是 x_j 上的振幅值。

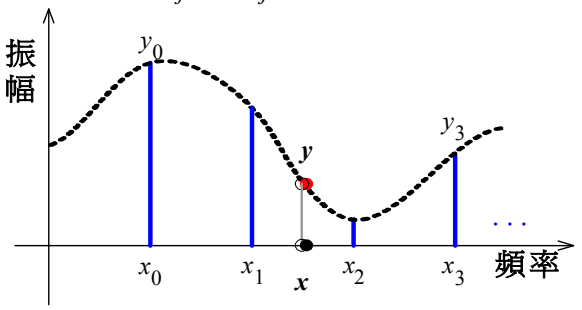

圖 2 振幅內差

在第一個音框上，要決定一個新的諧波頻率 x 的初始相位，我們也以 Lagrange 內插來先求出 x 對應的相位ϕ，然後依據公式(4)算出頻率 x 的相位增量，再倒退半個音框點數的相位量回去，即可求出 x 的初始相位θ。

2.3. 轉音、抖音處理

在國語歌曲裡，經常會有一個音節對應多個音符的情況，這樣的情況稱爲轉音。從歌聲轉音的聲紋圖(spectrogram)[18]，可明顯的看出轉音時，各個諧波的頻率是平滑地自前一個音符的頻率轉變至下一個音符的頻率。因此在轉音區段，我們必須機動地對各個諧波的頻率值及其對應的振幅值作調整，關於諧波頻率值的調整，考慮到聲紋圖上的變化必須保持平滑，所以不能夠將相位增量直接以線性內插來計算，我們的作法是，先將前後音符的各個諧波各自的相位增量都計算出來，進入轉折區段時，再把前後音符的對應諧波作相位增量的曲線式內插，在此我們使用了 cos 函數來作曲線內插。假設某一個音節含蓋了兩個音符，其中第 i 個音符的第 h 個諧波的相位增量爲 $\Delta\omega_h^i$，若兩音符之間的轉折區段的長度爲 M 個樣本點，則在第 m 個樣本點上的相位增量 $\Delta\omega_h[m]$，我們使用如下公式來計算：

$$\Delta\omega_h[m] = \left(\frac{\Delta\omega_h^1 + \Delta\omega_h^2}{2}\right) + \left(\frac{\Delta\omega_h^1 - \Delta\omega_h^2}{2}\right) \cdot cos\left(\pi \cdot \frac{m}{M}\right) \qquad (5)$$

使用此公式求出的相位增量值隨著時間改變的一個例子，如圖 3 所示。

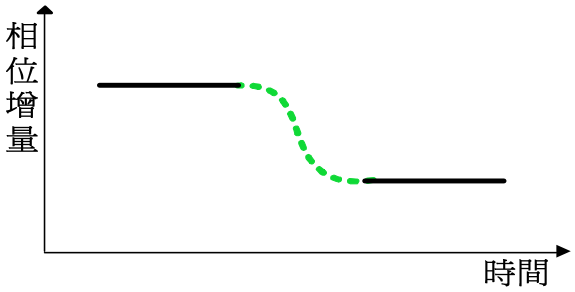

圖 3 曲線式內差之相位增量變化

在轉折區裡，各諧波的頻率值都會逐漸改變，所以各諧波的振幅值也必須跟隨頻率值的改變而改變，以保持音色的一致性。因此，對於第 h 個諧波，其振幅在第 m 個樣本點上的值 $A_h[m]$的求取，我們必須先依公式(5)算出時變之相位增量 $\Delta\omega_h[m]$，再用公式(4)反向求出對應的頻率值 $f_h[m]$，接著依此頻率值作 Lagrange 內插，以便求出時變的、可讓音色保持一致之諧波振幅值 $A_h[m]$。

人類歌聲還有一項特性，就是抖音，我們可從抖音信號的聲紋圖上，明顯地看到各諧波的紋路會如弦波一般地跳動。前面我們以公式(5)來變化相位增量的值，可以使聲紋圖上的一個諧波頻率值以曲線狀緩慢改變，同理我們也可用此作法來模擬抖音，只是要把變化的速度加快。設第 h 個諧波的相位增量爲 $\Delta\omega_h$，且抖音頻率爲 f_v（即每秒抖動 f_v 次），則抖音區段中第 m 個樣本點上的相位增量 $\Delta\omega_h[m]$ 的計算公式爲：

$$\Delta\omega_h[m] = \Delta\omega_h \cdot \left(1 + \lambda \cdot sin\left(2\pi \cdot f_v \cdot \frac{h \cdot m}{F_s}\right)\right) \qquad (6)$$

其中 f_v 的值通常介於 4.5~7Hz 之間[4]，而 λ 的值可設爲 0.2 左右。

3. MIDI 訊息處理

MIDI 檔案按照音軌(track)的個數以及 MIDI 訊息儲存方式的不同，共有三種格式，我們採用的是，各個聲道各自獨立儲存成音軌的 Format 1 格式。作 MIDI 檔分析時，我們系統會將每一個音軌的音軌名稱取出，作成一個選單，以供使用者選取主旋律所在之音軌。在使用者選好之後，系統接著把主旋律音軌上的訊息轉換成主旋律音符串列，再把 MIDI 檔中附隨的歌詞、或使用者輸入的歌詞轉換成歌詞串列，其它的 MIDI 訊息，則當作伴奏音符來整理成一個串列。

對於音符串列中的各個音符，我們所記錄的資訊

包含: (a)音高代號，(b)起始時間，(c)時長。音高代號可由 note on 訊息裡獲得，起始時間則依訊息前的 delta time 值去累計得到，而**時長**可由配對的 note on 與 note off 訊息，去相減各自所累計的時間值。此外，對於歌詞串列中的各個歌詞字，則記錄了起始時間之資訊。

有了主旋律音符串列和歌詞串列之後，再按照時間順序將兩者作結合，配對音符與歌詞，才能得到歌聲合成所需的音節演奏參數記錄(record)。一個音節演奏參數記錄中包含了一或多組的音高代號、以毫秒爲單位之時長、起始時間、及對應的音節參數檔名。

3.1. 歌詞資料

歌詞資料的來源可分爲兩種，一者是 MIDI 檔內附隨來的歌詞，另一者是使用者自己輸入的歌詞。當 MIDI 檔中沒有歌詞資料時，亦即沒有歌詞軌的 MIDI 檔案，我們系統就會依據主旋律音符串列，在每一個音符的位置，都對應出一個歌詞欄位，以供使用者輸入、編輯歌詞文字。

當歌詞編輯好後，我們系統可讓使用者操作 MIDI 檔的存檔動作。在 MIDI 檔中，歌詞資料是以 Meta-event 的方式來儲存，Meta-event 的格式爲<0xff, meta_type, v_length, event_data_bytes>，當 meta_type 爲 0x05 時，則表示 event_data_bytes 部分的資料是歌詞，歌詞資料的長度則記錄於 v_length 欄位。

3.2. 音符與歌詞之配對

關於主旋律音符和歌詞國字之間的配對，除了單一國字對應單一音符的情況之外，還必須考慮的一個情況是，國語歌曲裡經常會有單一國字對應多個音符(多組音高及音長)的情況，如圖 4 所示，這種情況就是所謂的轉音，例如兒歌"哥爸真偉大"的最後一句"只要我長大"，"要"對應了兩個音符。因此，音符串列和歌詞字串列，除了各自依時間次序作排序之外，我們還需要在兩者之間，作起始時間與音長的比對與結合。

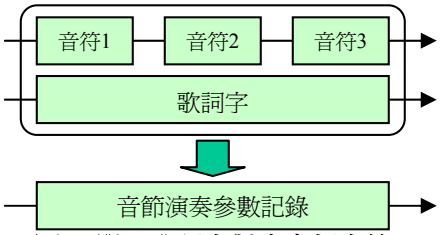

圖 4 單一歌詞字對應多個音符

依據歌詞 Meta-event 前的 delta time，雖然只能求得起始時間，但是我們可將相鄰歌詞之起始時間相減，而求得前一歌詞的時間長度。假設有一個歌詞訊息的起始時間與時間長度分別爲 T_y 與 D_y，而音符串列中第 i 個音符的起始時間與時間長度分別爲 T^i_n 與 D^i_n，在此考慮到兩者對應時，T_y 與 T^i_n 之間可能會有時間誤差，所以我們選擇可以使 $|T_y - T^i_n|$ 之值爲最小的第 i 個音符，作爲這一個歌詞訊息所對應的第一個音符。找到第一個音符之後，接下來就找出在這個歌詞資訊的時間長度範圍內，是否還有其他被對應到的音符。我們訂定公式(7)

$$T_y + D_y > T_n^{i+k} + D_n^{i+k} \tag{7}$$

來測試第 $i+k$ 個音符是否被對應到目前的歌詞訊息，如此繼續從第 $i+1$ 個音符起逐一比對，直到公式(7)不被滿足爲止。

3.3. 歌詞破音字處理

當我們將歌詞字與音符作配對之後，如果直接去作歌聲合成，很可能會發音錯誤，因爲有破音字的問題。例如在國語歌曲"特別的愛給特別的妳"裡，歌詞中有一句"特別的愛給特別的妳"，此句裡有兩個"的"，但前者發音爲/ㄉㄧˊ/，而後者發音爲/ㄉㄜ˙/。所以，必須對各個歌詞國字作檢查，以確定其是否爲破音字。

目前我們處理破音字的方式是，建立一個國字發音的對照表，據以查出各個歌詞字的所有可能發音，而讓使用者來選取正確的發音，即以手動方式解決破音字選音的問題。當使用者更改了破音字的發音之後，我們系統會將被選取的發音附加在歌詞國字的後面，並且在作 MIDI 訊息存檔時，也會把發音資料儲存起來，如此就可避免每次載入 MIDI 檔案都要再次選取發音的不便。

作歌聲信號合成時，我們系統會依歌詞的長度作判別，若歌詞長度超過 2 個 Bytes，表示國字後有附加發音，因此就取出附加的發音；若使用者沒有選取發音，則以歌詞國字所查出的第一個發音爲內定之發音。在確定一個歌詞字的發音之後，就可找出該發音所對應的音節信號參數檔，以取出所需的諧波頻率、振幅等參數來作歌聲信號合成。

4. 同步演奏之方法

在做完主旋律音符與歌詞的配對之後，我們系統本身會去合成各個歌詞字的歌聲信號，至於 MIDI 伴奏音符的聲音合成，目前我們系統不支援，而需交給作業系統去處理，一者由作業系統去作信號合成，或由作業系統送出給外接的 MIDI 合成器。在此情況下，我們必需考慮的一個重點是，如何讓合成的歌聲與 MIDI 伴奏音符作同步的播放，我們覺得一個可行

的方法是，將各個 MIDI 訊息及各個歌詞字所合成出的歌聲信號均視爲是帶有時間標記的事件，然後利用一個計時器來依時間順序送出各個事件給音效裝置(device)，而據以達到同步播放的目標。

關於前述方法的施行，有二個問題還需詳細考慮：(a)欲作到即時演奏，則歌聲信號的計算、與同步播放的控制，必須以並行(concurrent)處理的方式來進行；(b)計時器的精確性是必須顧慮、要求的。當合成歌聲信號時，若是等整首歌的音節都合成出來，才開始播放，則會讓使用者等待太久，因此我們採取兩個執行緒(thread)作並行處理的方式來進行，原先的執行緒負責 MIDI 訊息與合成歌聲的播放控制，另一個新起動的執行緒則負責歌聲信號的合成處理。

由於一個歌聲音節必須先合成好才能拿去播放，所以，播放的時間進度不能超過歌詞字合成出音節信號的處理進度。根據我們在 Pentium 1.8G Hz，256MB 記憶體的電腦上所做的實驗，合成播放時間爲一秒的一個諧波所需要的時間約爲 14.5 毫秒，也就是說在一秒內，我們最多能夠合成 1,000 / 14.5 = 68.9 個諧波，所以在最緊湊的情況，合成處理的時間長度等於播放的時間長度，一個歌聲音節的基頻，最低可以低到 11,025/68 = 162.1Hz。由於我們錄的音源音節是女聲，且平常設定的歌唱聲基頻大都是從 C4 音高(約 261Hz)左右開始，所以可以確定合成所需的時間會比播放的時間來得短，因此只需檢查歌聲是否有從時刻爲零的時間點就開始播放的，如果有此種情況，則也只需等合成好第一個音節之後，就可以開始播放了。

在計時器的部分，最初我們是用 BCB(Borland C++ Builder)裡的 TTimer 元件，來作計時及事件的觸發，但是它的精準度不高，每一秒鐘最多只能發出 18 次 timer 中斷(interrupt)的訊息，因此造成整首樂曲聽起來速度忽快忽慢，節拍整個錯亂，再者從 MIDI 時間點數(ticks)換算成時間，也會導入些許的誤差，因此，我們必須使用一個高精準度的計時器，才不會在播放時發生節拍錯亂的情況。

在研讀 Windows 作業系統的文獻之後，我們找到了 Win32 API 中的多媒體計時函式(multimedia timer functions)，用它來實施高精確度的計時功能。作同步播放控制的執行序，實際上就是依照圖 7 的流程來動作，首先在「開啓計時器」裡，先呼叫 timeGetDevCaps( ) 函式，將計時器精準度設定爲 1 毫秒後，再呼叫 timeBeginPeriod( )函式來開始計時，開始計時之後，就可呼叫 timeSetEvent( )函式來設定毫秒時間尺度所要觸動執行的函式，即在 timeSetEvent( ) 的參數中要指定一個 callback 函式，所以在我們的系統中，這一個 callback 函式會執行時間的比對，及送出 MIDI 訊息或音節信號去播放。

我們系統對於 MIDI 訊息的處理是，呼叫 midiOutShortMsg( )函式去播放，而對於音節信號，則是呼叫

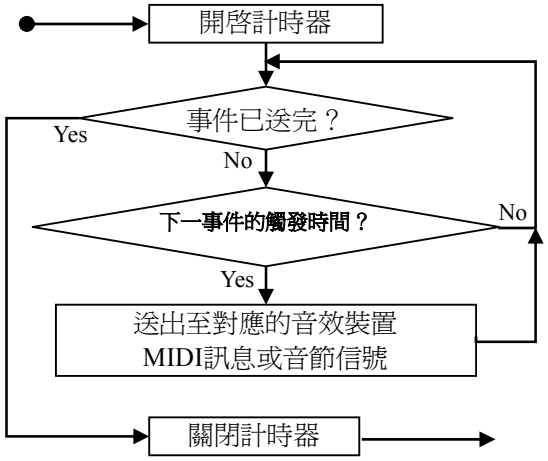

圖 5 同步播放之處理流程

waveOutWrite()來播放。另外需注意的是，在結束播放處理時，必須呼叫 timeKillEvent( )與 timeEndPeriod( ) 兩函式，來將計時器關閉，否則系統的反應會變得越來越慢。

5. 聽測評估

這裡考慮了三種國語歌曲合成的方式：(a)TIPW 合成法[12]、(b)本文的加式弦波合成法、(c)本文方法合成出的歌聲再加上 MIDI 伴奏。所使用的歌曲爲兒歌"只要我長大"，評估的項目有三項，分別是清晰度、自然度、及喜好排序。清晰度主要是評估合成出來的歌聲信號聽起來是否清楚無雜訊，以及咬字的清楚程度，自然度則是評估合成歌聲與人聲的接近程度，喜好排序則是將前述三種合成方式(a)、(b)、(c)，依照喜好程度由高至低排列。評分的標準依照優、佳、可、差、劣五個等級分成 5、4、3、2、1 等級分。

測試者的性別爲 6 位女性及 9 位男性；年齡層 20 歲至 30 歲共 14 位，30 歲以上 1 位。聽測的方法是，隨機排列三種合成方式所合成出的歌聲，讓測試者試聽及評分，評分的平均值結果如表 1 所示。

表 1 聽測實驗之結果

	清晰度	自然度	喜好排序
TIPW 無伴奏	**3.73**	**3**	**3**
弦波模型 無伴奏	**4.13**	**4**	**1.87**
弦波模型 MIDI 伴奏	**4.46**	**4.46**	**1.13**

由表 1 的結果可發現，弦波模型所合成出的歌聲比起 TIPW 法合成出的歌聲，其自然度提升了 1 個等級，而清晰度也有些許的提升，自然度提升的原因，

應是由於以弦波模型來作歌聲合成，並且加入抖音及轉音，可以比較接近真人唱歌時的感覺，因此聽起來較爲自然；而清晰度的提升，應是由於使用本論文的加式弦波模型作合成，聲音較爲清亮，且對於咬字的清楚程度也有幫助。

就弦波模型來看，加上 MIDI 伴奏比未加伴奏的歌曲，清晰度與自然度分別上升了 0.33 與 0.46 級分，這顯示有 MIDI 伴奏的樂曲對於人類聽覺而言，可以使合成歌曲聽起來更逼近真人所唱的感覺。

6. 結論

我們以加式弦波模型爲基礎，研究國語歌聲合成的方法，獲得的成果包括: (a) 以 Lagrange 內差來求出頻率移動後的諧波的振幅值，因而能夠維持音色的一致性；(b)提出歌聲裡轉音與抖音效果的合成方法；(c)提出音節子音、母音的時長調整方法。此外，我們也提出了一種作法，來讓國語歌聲和 MIDI 伴奏音符作同步、即時之演奏，亦即採取多執行序之設計和使用較爲精準的計時器。爲了驗證所提出方法的效能，我們實際製作了一個整合 MIDI 伴奏之國語歌聲合成系統，執行後的一個畫面如圖 6 所示，它可依 MIDI 檔裡的主旋律音符和歌詞訊息，來合成出國語歌聲，再和 MIDI 伴奏音符作同步、即時之演奏。經由聽測實驗的評估結果可知，合成的國語歌聲配上 MIDI 伴奏，的確可在自然度、清晰度、及喜好度方面，比未加上伴奏的清唱式合成歌聲好很多。

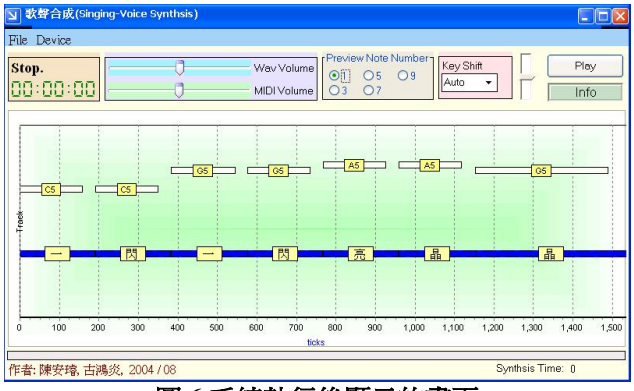

圖 6 系統執行後顯示的畫面

未來我們可再繼續研究、改進的方向如: (a)相鄰音節之間仍有分離感，原因之一是以線性比例作時長之調整，未來可嘗試改成 ADSR 方式[10, 11]；(b)相鄰音節之間的聲紋移轉，會有不連續的現象，造成短音長的音節，不能平順地銜接；(c)目前只作到單聲部歌聲信號的合成，未來可再研究多聲部歌聲的合成處理，以讓合成之國語歌聲有更豐富的表現。

7. 參考文獻

[1] Cook, R., "Singing Voice Synthesis: History, Current Work, and Future Directions", *Computer Music Journal*, Vol. 20, No. 3, pp. 38-46, 1996.

[2] George, B., Y. Ding, and S. Yim , "Fixed frame rate ABS/OLA sinusoidal modeling applied to polyphonic music synthesis", *IEEE First Workshop on Multimedia Signal Processing*, pp. 59-64, June 1997.

[3] Macon , M.W., L. Jensen-Link, J. Oliverio, M. A. Clements, and E. B. George, "A Singing Voice Synthesis System Based on Sinusoidal Modeling", *IEEE ICASSP-97*, Vol. 1, pp. 435-438, 1997.

[4] Meron, Y. and K. Hirose, "Synthesis of Vibrato Singing, " *IEEE ICASSP '00*, Vol. 2, pp. II745-II748, 2000.

[5] Lee, M. E. and M. J. T. Smith, "Spectral Modification for Digital Singing Voice Synthesis Using Asymmetric Generalized Gaussians", *IEEE ICASSP-03*, pp. I260-I263. 2003.

[6] 邵芳雯，國語歌曲之合成，碩士論文，國立交通大學電信研究所，1994。

[7] 林政源，國語歌曲的歌聲合成，碩士論文，國立清華大學資訊工程研究所，2001。

[8] 盛思豪，即時歌唱聲合成系統與音樂合成系統之整合，碩士論文，國立台灣科技大學電機研究所，2002。

[9] 歐婉菁，合成歌聲，碩士論文，國立臺灣大學資訊工程研究所，2003。

[10] Dodge, C. and T. A. Jerse, *Computer Music: Synthesis, Composition, and Performance*, 2'nd ed., New York: Schirmer Books, 1997.

[11] Moore, F. R., *Elements of Computer Music*, Prentice-Hall, 1990.

[12] Gu, H. Y. and W. L. Shiu, "A Mandarin-syllable Signal Synthesis Method with Increased Flexibility in Duration, Tone and Timbre Control, " Proc. National Science Council, R.O.C., Part A: *Physical Science and Engineering*, Vol. 22(3), pp. 385-395, 1998.

[13] 古 鴻 炎 ， http://guhy.csie.ntust.edu.tw/syn-sound.html (可試聽合成的國語歌聲)，國立台灣科技大學資訊工程系.

[14] 李雪貞，客語語音合成之初步研究，碩士論文，國立台灣科技大學資訊工程研究所，2001。

[15] 陳安璿，整合 MIDI 伴奏之歌唱聲合成系統，碩士論文，國立台灣科技大學資訊工程系，2004。

[16] Roads, C., *The Computer Music Tutorial*, MIT Press, 1996.

[17] Stoer, J. and R. Bulirsch, *Introduction to Numerical Analysis*, 2'ed., New York: Springer-Verlag, 1993.

[18] O' Shaughnessy, D., *Speech Communications: Human and Machine*, 2'ed., IEEE Press, 2000.