声乐合成初探

王赟

（清华大学 电子工程系 北京 100084）

摘要

本文首先扼要地介绍了当前语音合成的三种主要途径，然后具体介绍了作者实现拼接法声乐合成的实验过程。对于实现中遇到的部分问题提出了解决方案，并提出了未来工作的几个方向。

关键词

声乐合成，语音合成，拼接法，频率测定，频率调整，时长调整

Preliminary Attempt at Vocal Music Synthesis

Wang Yun

(Department of Electronics Engineering, Tsinghua University, Beijing, 100084)

Abstract

In this paper, first I introduce the three main methods of voice synthesis at present in brief, then I introduce the procedure of my attempt at implementing concatenative vocal music synthesis. Solutions are proposed to some of the problems encountered in the procedure of implementation, and several fields of future work are proposed.

Keywords

Vocal music synthesis, voice synthesis, concatenative method, frequency measurement, frequency modification, duration modification

1. 引言

2003 年 12 月，日本 Yamaha 公司推出了一款名为 Vocaloid 的软件，这是世界上第一款专门实现声乐合成的软件。它对于专业的音乐人士有很大帮助，对于普通的电脑用户，也提供了一种全新的娱乐方式。

声乐合成是语音合成的特殊情形。自 1779 年世界上制造出第一台有记载的语音合成器以来，语音合成已有 200 多年的历史。1791 年，von Kempelen 制造出第一台能把元音、辅音同时合成的声音合成器，据说它能够合成由 19 种辅音和 5 种元音组合的单词。[4]在

当代，计算机的发展为语音合成提供了很大的便利。目前语音合成大致有拼接法、规则合成法以及声道模拟法三种。

本文作者使用拼接法实现了一个声乐合成系统，利用预先录制的原始语音材料，经过材料分割、频率测定、频率及时长调整、连接与混音等步骤合成声乐。下面将介绍语音合成的三种途径及拼接法声乐合成的实现过程。

2. 语音合成的三种途径及比较

2.1 拼接法

拼接法的思路是预先录制原始语音材料，然后将它们编辑、连接。

原始语音材料可以取不同的单位。比较简单的方法是以音素或音节为单位。为了使不同单位的连接更加平滑，还可以选用更大的单位，如双音子、三音子等。双音子(diphone)是两个连续发音的音素中，从前一个音素的正中间到后一个音素的正中间的部分；三音子(triphone)则是三个连续发间的音素中，从第一个音素的正中间到第三个音素的正中间的部分。选取的单位越大，合成语音的连续性就越好，但所需的工作量及存储空间也越大。比利时的 MBROLA 项目[7]是研究世界各国语言的语音合成的，他们采用的就是双音子数据库。其中一个法语男声的双音子数据库花费了约一个月的时间录制，占用了 4.4MB 的存储空间。而录制音节数据库则简单得多，往往只需几个小时。

拼接法由于其实现较简单，主要依赖大容量数据库，因此在大容量存储设备唾手可得的今天取得了较大的成功，其不少产品已经商业化。

2.2 规则合成法

规则合成法的思路是分析各种音素的波形，用参量对之进行数学的描述，通过调整参量来合成不同的语音。

规则合成法需要的存储空间很小，但由于它完全通过数学的方法合成语音，因此在自然性方面比不上拼接法。

2.3 声道模拟法

声道模拟法的思路是模拟人的声道发声时的物理特性，由此计算口、鼻处的气体体积速度及压力，并合成语音。

该方法通过 X 射线等方法获得发声时声道的剖面图，把它近似地看作一串共轴圆柱。每个圆柱两端的气体体积速度及压力有一定规律，可用一个矩阵表示。这样，在测定出发音处的气体体积速度及压力后，对于非鼻音，可以通过一串矩阵相乘求得口处的气体体积速度及压力；对于鼻音，则声道有分支，模型会稍微复杂一些，但思路是相同的。

声道模拟法与前两种方法相比，采用的是完全不同的途径。前两种方法都是只研究输出，即所谓的“黑箱法”，而声道模拟法则研究声音产生的过程。这种方法是近几年刚刚兴起的，目前的成功程度比不上拼接法，但专家预计这种方法最有前途。

3. 声乐合成与语音合成的比较

一般的语音合成的步骤是比较复杂的。它首先需要把文本转化成语音符号，即对文本注音，这需要一个庞大的词典。此外，还要确定断句的位置以及音高的高低等信息。最后根据上面所得的信息合成语音。

声乐合成则比此简单得多。断句的位置、音高的高低等信息完全由乐谱决定，而歌词则可以直接输入读音而不是文本，这样声乐合成的前两个步骤就可以省略了。而且由于声乐中对相邻两个音节的连续性的要求不是很高，因此在采用拼接法时，完全可以以音节为单位。

4. 拼接法声乐合成的实现过程

4.1 语言的选择

在实际的软件开发中，语言的选择似乎不应该成为一个课题。但由于本文所述的是一个实验，为了在达到效果的前提下尽可能减小工作量，这还是一个值得研究的问题。

本文采用的是拼接法，需要录制各个音节的发音。为了最小化工作量，当然应当选择音节数少的语言。各种欧洲语言的音节基本都可以以辅音结尾，而且元音数量又很多，音节数成千上万；而且由于结尾的辅音发音较轻，在下面的切割步骤中容易被切去，故不适宜用作实验的材料。我所知道的比较适宜作实验材料的语言有汉语普通话和日语，前者约有 400 个音节；而后者由于元音个数极少（仅有 a,i,u,e,o 五个），常用音节只有 102 个（促音由于在歌曲中无法唱出来，不计），即使加上用于拼写外来语的极不常用的音节，也不超过 200 个。

另外，在设计乐谱时，为简便起见，采用了 txt 格式。这需要把歌谱和歌词对齐，所以需要每个音节可以用尽可能短的字符串表示。日语的音节都可以用 1~3 个字母表示；中文尽管可以用汉字，且一律占 2 个字节，但中文有大量的多音字，给它们注音的工作量甚至会超过声乐合成本身。使用双拼编码也可以保证每个音节仅用 2 个字母表示，但这样可读性差。

综上，本实验选用日语作为实验材料。

4.2 数据库的建立

4.2.1 RIFF WAV 文件格式简介[5]

本文实验采用的是 wav 文件，其格式称作 RIFF WAV。

RIFF WAV 格式文件的头部为 12 个字节，内容如下表所示：

偏移量	内容及含义
0~3	四个字符 RIFF
4~7	一个无符号长整型数，表示此后文件的字节数，等于文件总字节数减 8
8~11	四个字符 WAVE

接下来有若干个块(chunk)。每个块的构成如下：

偏移量	内容及含义
0~3	四个字符，表示块的名称
4~7	一个无符号长整型数 ChunkSize，表示当前块剩余部分的字节数
8~ChunkSize+7	块的内容

RIFF WAV 格式中常用的块有如下几个：

名称	内容	备注

"fmt "	文件格式 说明	1. 注意名称的结尾有一空格 2. 该块不可省略，且必须是第一个块
"fact"	总帧数	1. 帧的英文为 block，含义见下文。这样译是为了防止与 chunk 混淆。 2. 该块可以省略，若不省略，必须位于"data"块之前 3. [5]中该块的内容作“样本总数”，疑误
"data"	波形	该块不可省略
"LIST"	注释及版权信息等	1. 块名是大写 2. 该块可以省略

其中，本实验仅用到"fmt "和"data"两个块。

"data"块的内容就是波形数据，该数据是通过采样获得的。按一定的速率（称为“采样率”）记录声波的位移，就可以还原出声音。常用的采样率有 44100Hz、22050Hz、11025Hz 等。一般没有比 44100Hz 再高的采样率，因为人耳能听到的最高频率约为 20000Hz，根据信息论中的“仙农采样定理”，只要采样率达到这个频率的 2 倍即可保证不失真[6]。对于单声道文件来说，每次采样可以得到一个位移；对于立体声文件则可以得到左、右声道的两个位移。每个位移称为一个“样本”，这一个或两个样本称为一个“帧”。每个样本的表示方法有两种：一种是用 8 位无符号数表示，128 表示无声；一种是用 16 位带符号数表示，0 表示无声。"data"块的内容就是按时间顺序记录的各个样本的数值，同一帧中若有两个样本，则按左、右声道的顺序排列。

"fmt "块的内容可以看成一个结构体：

```
struct {
  short FormatTag;
  unsigned short ChannelCount;
  unsigned long SamplesPerSec;
  unsigned long BytesPerSec;
  unsigned short BytesPerBlock;
  unsigned short BitsPerSample;
}
```

其中，FormatTag 表示文件的类型，默认为 1；ChannelCount 为声道数，单声道为 1，立体声为 2。以下四个变量实际提供的是比特(bit)、字节(byte)、样本(sample)、帧(block)、秒(second)各单位之间的换算关系，其中是有冗余的。这个关系可以用图 1 表示。图中加下划线的量是"fmt "块中已经提供的，无下划线的量是可以通过加下划线的量算出来的，以后直接应用。此外，"data"块的 ChunkSize 就是波形的总字节数，通过它可以换算出波形的总帧数 BlockCount 以及总时长等。

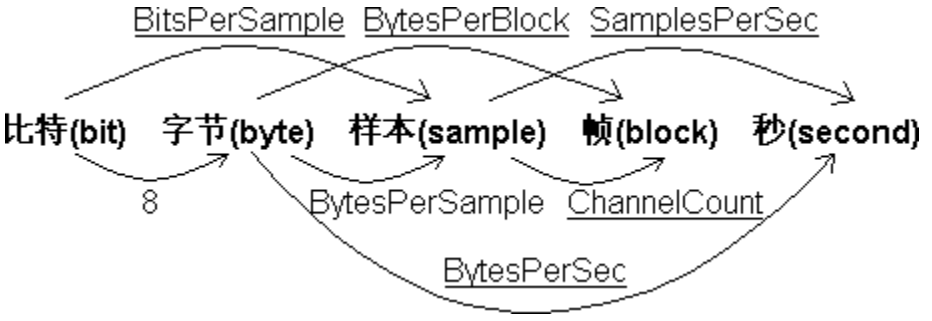

图 1 各单位的换算关系

需要说明的是，"fmt "块中在上述数据之后还可能包含其它内容，本实验中忽略。

4.2.2 原始语音材料的录制

原始语音材料指的就是选定的语言中所有音节的发音。录制过程用 Windows 自带的附件程序“录音机”就可以完成。用一些专门的音频处理软件，如 Cool Edit，则可以监视录音的质量，效果会更好。由于内存的限制，我把日语中的 102 个音节分成了三类（见表 1），分别录制成三个文件。一些特殊音节，如 we,fa 等，由于很不常见，可以在用到的时候当场录制。

文件 1 （五十音图）	文件 2 （浊音、半浊音）	文件 3 （拗音）
a i u e o	ga gi gu ge go	kya kyu kyo
ka ki ku ke ko	za ji zu ze zo	sha shu sho
sa si su se so	da de do	cha chu cho
ta ci cu te to	ba bi bu be bo	nya nyu nyo
na ni nu ne no	pa pi pu pe po	hya hyu hyo
ha hi fu he ho		mya myu myo
ma mi mu me mo		rya ryu ryo
ya yu yo		gya gyu gyo
ra ri ru re ro		ja ju jo
wa wo		bya byu byo
n		pya pyu pyo

表 1 日语音节的分类

为了在基本保证音质的前提下尽可能减小文件的体积，我选择的"fmt "块参数为：采样率 11025Hz，样本用 8 位无符号数表示，立体声，即 SamplesPerSec=11025, BitsPerSample=8, ChannelCount=2。在朗读各个音节时，应注意音高不要起伏，音节与间隔的时间稍长，辅音要读得短促而清晰、响亮。

4.2.3 原始语音材料的切割

在录制时，把许多音节都录制到一个文件里是为了操作简便，但最终合成时使用的应当是许多只包含一个音节的文件。而且，即使在录制时把每个音节都录制成单独的文件，还需要去掉每个文件开头和末尾的空白。所以，有必要对原始语音材料进行切割，提取出前后不带空白的每个音节。

下面以单声道为例说明切割的过程。切割过程用到了四个参数：MinUnitDuration, MinRestDuration, MaxRestOffset, MinRestOffset，分别表示音节的最短时长、间隔的最短时长、间隔中波形的最大和最小位移。调整这四个参数，就可以达到对噪音的不同容忍程度。根据"fmt "块中的参数，可以由 MinUnitDuration 和 MinRestDuration 算出音节和间隔的最小帧数 MinUnitBlocks 和 MinRestBlocks。从头到尾扫描整个波形的每一帧，若遇到一个位移超出了区间[MinRestOffset, MaxRestOffset]范围的帧，则假设找到了一个单元的开始；然后继续扫描至遇到一个帧数不小于 MinRestBlocks 的间隔，即连续 MinRestBlocks 个帧的位移都在区间[MinRestOffset, MaxRestOffset]之内，则认为找到了这个单元的结束。若此单元的帧数不小于 MinUnitBlocks，则认为这是一个音节，存到一个单独的文件中去，否则将此单元舍弃。

对于双声道，则每个声道有各自的 MaxRestOffset 和 MinRestOffset 值。单声道的情况中“位移超出[MinRestOffset, MaxRestOffset]区间”的条件改为“至少有一个声道的位移超出了[MinRestOffset, MaxRestOffset]区间”。

4.2.4 各音节频率的测定

频率的测定是数据库建立过程中至关重要的一环。即使不需要进行频率调整而仅仅是调整时长，也需要对频率进行测定。其原因将在 4.3.1 节讲述。

但是，频率的测定也是一个比较困难的。由于人声是由多种不同频率的波复合而成的，在测定中就容易捕捉到一些比较强的、频率与待测频率成整数倍关系的波。为简单起见，本文所述实验在测定各音节频率之前预先估计了一个频率范围（即引入两个参数 MinFreq, MaxFreq），如男声的频率一般在 100~300Hz 之间，对于某个具体的音高，则可以有更精确的估计。这样就巧妙地回避了测量值与真实值不相等而是成整数倍关系的问题。如果预先无法估计频率的范围，则可以用一个较广的范围去测定大量的样本，如所有音节，观察大多数测量值落在哪个较小的区间内，以此区间作为估计频率范围。

在有了估计频率范围之后，为了精确测量频率，我取了第三个参数——频率步长 StepFreq。在估计频率范围内每隔 StepFreq 取一个频率，求出周期，再求出一个周期的帧数。在整个音节上等间隔地取 MeasurePointCount 个采样点（这是第四个参数），分别测定这些采样点附近的频率。在每个采样点处的一个周期内等间隔地取 SamplesPerPeriod 个采样点（这是第五个参数），计算这些采样点与一个周期之前及一个周期之后的波形位移的差的绝对值，并累加。累加和最小的频率就认为是被测采样点的频率。取各采样点频率测量值的中位数作为被测音节的频率。之所以取中位数而不是平均数，是因为该算法一旦产生实质性的偏差，测量值就会是真实值的整数倍或整数分之一，这个误差是巨大的，而平均数易受极端数据的影响，取平均数往往会使结果毫无意义。而取中位数则可以避免极端数据的影响。

由于波形是离散采样的，上面的计算过程中会遇到两帧之间某个时刻的波形位移，我采用的是最简单的线性模型，即认为相邻两帧之间的波形是线段。

上面这种方法在限定了一个确实包含待测频率真实值的频率范围之后，测定的准确率基本可以达到 100%。在没有限定范围（即范围过宽，如 100~2000Hz）的情况下，测定的准确率与录音质量有关，一般至少可以达到 60~70%。

4.2.5 各音节其它信息的获取

其它信息包括音节的时长和平均波形位移。

时长是很容易计算的，只需用总帧数除以 BlocksPerSec 即可。但实际存到数据库中的时长仅为实际时长的 99%，这是为了防止实数取整时产生误差而使得程序读取的帧数超过了音节的实际帧数。

平均波形位移的计算也很容易。之所以要计算这个信息，是因为我注意到在录音时，间隔部分的波形并不是总接近“无声位移”（8 位为 128，16 位为 0）。如果一律以“无声位移”为平均值，那么在音节的音量需要放大时，“平均位移”与“无声位移”的差（称为“直流偏移”）所产生的噪声也会被放大，影响合成音乐的质量。计算了“平均位移”之后，这部分噪声就可以基本消除。

4.2.6 数据库建立过程的自动化

经过上面的过程，我们就获得了一个音节的全部信息。然而，上面的过程中引入了太多的参数，而这些参数大多是在屡次试验中找到满意的值的。由于音节数较多，对每个音节都进行手工操作，工作量将大得可怕。因此有必要实现自动化。

自动化的思路就是固定一些参数，让少量参数自动变化。“少量”的要求是每个过程中只能有一个参数在变化。否则，若要保证精确性，枚举量就不可承受。我的程序中是这样实现自动化的：

在切割阶段，预先求出包含很多音节的波形的每个声道的最大、最小及平均位移，并以最大位移与平均位移的差及平均位移与最小位移的差中的较大值作为该声道的振幅。这一阶段中 MinUnitDuration 及 MinRestDuration 两个参数都固定为 0.1 秒。本阶段中设一个变量 NoiseProportion，以 0.01 为步长从 0.2 变化至 0（上下限可调整）。对于一个确定的 NoiseProportion，每一个声道的 MaxRestOffset 及 MinRestOffset 可由式子 Average ±Amplitude * NoisePropotion 算得，其中 Average 及 Amplitude 分别表示这个声道的平均位移和振幅。之所以让 NoiseProportion 递减而不是递增，是为了尽可能排除掉音节首尾的噪声。

在频率测定阶段，MinFreq 与 MaxFreq 通过文件输入，StepFreq, MeasurePointCount, SamplesPerPeriod 分别取 1, 5 和 50。这些参数兼顾了精确性和效率。

经过自动化，建立数据库的整个过程就可以由一个程序来完成了。它的输入是每个原始语音材料的文件名、对应的音节表所在的文件名以及估计的频率范围。它的输出是由每个音节的信息组成的数据库，在声乐的最终合成中作为输入。每个音节的信息可以组织成一个结构体：

```
typedef struct {
  char name[4]; //音节名
  double duration; //时长，为实际时长的 99%
  double consonant; //辅音时长（这个域是后加的，详见第 5 节）
  double freq; //频率
  double average[2]; //每个声道的平均位移
  char file[32]; //该音节所在的文件名
} SYLLABLE;
```

数据库中存放的就是这种结构体组成的数组。

4.3 声乐的最终合成

4.3.1 音节频率与时长的调整

音节频率与时长的调整，就是要把录制的时长为 D0、频率为 F0 的音节调整成时长为 D1、频率为 F1 的音节。其中 D0、F0 是测定出来的；D1 与 F1 是由乐谱得到的。

先看比较简单的频率调整。由于我们最终要进行时长的调整，故在频率调整中可以允许时长有变化。在这样宽松的条件下，只需对原来的波形换一个采样率重新采样，但仍按原采样率存储即可。令新采样率为原采样率的 F0/F1 倍，即可将音节的频率提高到原来的 F1/F0 倍，同时时长变为原来的 F0/F1 倍。频率调整过程中也会遇到需要求两帧之间某时刻波形位移的问题，这里同样采用最简单的线性模型。

然后再看时长的调整。最初我朴素的想法就是，由于音节的中部是元音，它的波形基本是重复的，把音节中部的一段重复若干次或删去，使得时长变为 D1 即可。但通过观察波形发现，一个音节的振幅随时间有缓慢递减的趋势，如图 2，如果直接大段复制或删除，这种趋势就会被破坏，影响合成音的自然性。

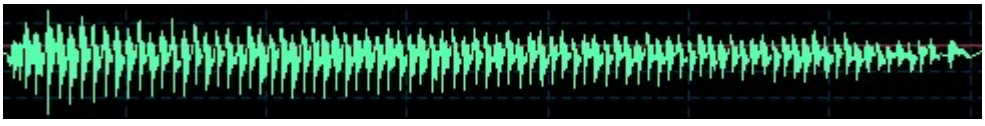

图 2 频率约为 130~135Hz 的男声"a"的波形。
从图中可以看到振幅有缓慢递减的趋势。

因此最终我采取的是这样的算法：设时长调整前后的周期数分别为 P0 和 P1，则调整后的第 P 个周期采用的是调整前第 P/P1*P0 个周期的波形。若 P/P1*P0 不是整数，则取其整数部分。这里是取整还是四舍五入是不重要的。在此可以解答前文留下的“为什么时长调整也需要测定频率”的问题：计算每个周期的长度和总周期数是需要知道频率的。

这样，把时长为 D0、频率为 F0 的音节调整成时长为 D1、频率为 F1 的音节的操作过程就是：先把时长调整为原来的(D1/D0)*(F1/F0)倍，再把频率调整为原来的 F1/F0 倍。

4.3.2 乐谱文件的格式

考虑到编辑和程序读取两方面的方便，我采用了 txt 格式的乐谱文件。其规定如下：

乐谱文件由命令和乐谱构成。

每条命令占一行，命令为一个单词（大小写敏感），其后可能有参数。目前支持的命令有：

1) Singer [参数]，表示歌曲是由谁唱的，即使用哪个数据库。
2) Key [参数]，表示歌曲的调式。由主音的音名与一个数字构成。音名为一个 C,D,E,F,G,A,B 中的某一个大写字母，前面或后面可以加一个'#'或'b'表示升降半音。数字表示主音处在哪一个八度，数字增加 1，就升高一个八度。以 A4 为国际标准音 6（440Hz）。
3) Beat [参数 1]/[参数 2]，表示歌曲的拍号。
4) Tempo [参数]，表示歌曲的速度为每分钟多少个四分音符。
5) Quantize [参数]，表示乐谱中每两列代表一个几分音符。
6) AccomAmplify [参数]，表示伴奏音量的放大倍数。
7) VoiceAmplify [参数]，表示人声音量的放大倍数。调整这两个参数可以解决伴奏

与人声音量悬殊的问题。

8) RestTime [参数]，表示休止一定时间，单位为秒。多用于跳过歌曲开头的空白与前奏。

9) RestBars [参数]，表示休止若干小节。多用于跳过间奏。

乐谱是由两行 Melody 与 End Melody 括起来的部分，四行为一组。每组中前三行为曲，第四行为词。每两列表示一个 x 分音符，其中 x 是 Quantize 命令设定的。这两列中，第二行第 1 列为唱名，第一行第 1 列可以有高音点'.'或':'，第三行第 1 列可以有低音点'`'或':'，第四行为音节的名称。唱名 1,2,3,4,5,6,7 分别用对应的数字表示，升半音的 1,2,4,5,6 分别用字符'!','@','$','%','^'（即 Shift+数字）表示，0 表示休止符。若唱名为空格，则表示延长前面的音符。若音节为三个字符，则可在前三行的第 3 列均填入字符'&'，程序可以知道此处为三列共同表示一个音节。音符的时长不需要用特殊的符号表示，而是由后面有多少空列表示。乐谱部分出现的上述字符以外的字符都将被忽略，因此可以用'|'表示小节线而增加可读性。乐谱的示例可以参看附录。

乐谱文件中 Melody 与 End Melody 之外的、不是任何命令的行将被忽略。因此可以在乐谱范围之外自由地添加注释。

4.3.3 结果 WAV 文件的生成

这一步的技术性就比较弱了，主要是编程的问题。这一步所要做的就是根据乐谱得出需要把哪个音的时长和频率调整到多少，调整后混合入伴奏文件的适当位置。我的程序要求伴奏文件与音节文件的"fmt "块参数相同。

此步骤中需要按照乐谱文件中的命令调整伴奏和音节的音量并混合。可通过如下公式计算混合后各声道的波形位移：

```
Offset = (AccomOffset - Silence) * AccomAmplify
       + (VoiceOffset - VoiceAverage) * VoiceAmplify + Silence
```

其中，Offset 为混合后的位移，AccomOffset, VoiceOffset 分别为伴奏与音节的位移，AccomAmplify 与 VoiceAmplify 为乐谱文件中设定的伴奏与音节的放大倍数，VoiceAverage 为音节的平均波形位移，Silence 为“无声位移”，8 位时为 128，16 位时为 0。之所以没有求伴奏的平均波形位移，是因为伴奏的音质一般比较好，不存在直流偏移。若求出的 Offset 超出了允许的位移范围（8 位为 0~255，16 位为-32768~32767），则令其为最大或最小允许位移。

对于圆滑音（即一个音节对应多个音符的现象），我是这样处理的：第一个音符正常处理，以后的每一个音符，采用该音节的元音对应的音节来合成，如圆滑音 ka 的第二个音符实际是用 a 来合成的。

5. 实验过程中遇到的几个问题及解决方法

在实验的过程中，我遇到的问题主要有如下两个。

第一个问题出现在频率调整中。当频率调整的范围较大（达到半个八度）时，音色会发生明显的变化。若频率升高，则男声会变成女声，女声会变成童声，使得歌曲的感情色彩受到很大影响。而若频率降低，则常常根本分辨不出唱的是哪个音节。

上面的现象说明同一个音节在音高不同时，不仅频率不同，一定还有别的差别。通过查看波形发现，不同音高的同一个音节的波形就非常不同，如图 3。

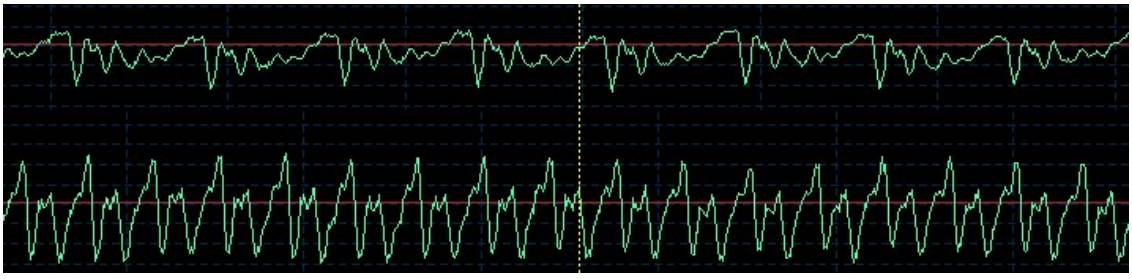

图 3 不同音高的同一音节的波形差别
上：频率约为 130~135Hz 的男声"a"的波形
下：频率约为 260~270Hz 的男声"a"的波形

由此可以看出，只由一个特定音高的音节通过频率调整获得自然性很好的各种音高的音节是很困难的。因此我只好采用一个折衷的方案：录制若干套不同音高的音节表，根据实际需要合成的音高，选择适宜音高的原始音节。考虑到频率降低后容易导致音节难以辨认，故“适宜”的含义定为：不超过待合成音高的最高音高。

为了在尽可能短的时间内验证上述方案的可行性，我没有录制自己的声音，而是从 MBROLA 项目的网站[7]上下载了一个名为 Tachiki 的合成女声，用 Tachiki 朗读的不同音高的音节作为原始语音材料构建了数据库。所采用的不同频率有 110Hz, 220Hz, 330Hz, 440Hz, 550Hz 和 660Hz。用这种方法合成出的声音比用单一频率合成的效果好得多。在此鼓励下，我又录制了自己的四种音高（频率分别约为 130Hz，164Hz, 207Hz, 261Hz）的声音作为原始语音材料，合成效果也有了明显改善。

第二个问题出现在时长调整中。当音符很长时，音节中的辅音也会被显著拉长，使得听起来感觉人声比伴奏慢了一段时间。辅音为 m,n 等鼻音或 s,w 时这种现象尤其明显。对此，我修正了时长与频率调整的方法。我假定每个音节开头长度为 consonant 的时长段为辅音，对这一段只调整频率而保持时长不变，而对后一段同时调整时长和频率。consonant 值难以测定，我采用的如下的权宜之计：

- 对于辅音是摩擦音（s,z,j,c）的音节，consonant=0.2s；
- 对于其它音节，consonant=0.1s；
- 若上面的值超过了音节总时长的一半，则改为音节时长的一半。

第三个问题是圆滑音中除第一个音符外，后面的音符的开头有时显得比较突兀。我采用的方法是：圆滑音中除第一个音符以外的音符，使用对应音节的元音部分进行合成。当然，这种方法可能只适用于日语这种只有开音节，且只有单元音的语言，其它语言中像 kai, kang, kap 这类有复元音和闭音节的情况应该更复杂。

6. 将来的工作

针对实验中遇到的问题，我认为可以从以下几个角度考虑以提高合成声乐的质量：

1. 研究如何根据一个特定音高的音节获得自然性很好的各种音高的音节。
2. 设法自动化地测定出每个音节中辅音的时长。

以上两点将是比较有难度的，尤其是第一点。

另外，在实验的过程中，我感到这个系统使用起来还不是很方便。主要不便之处在于录音的工作量太大，以及输入乐谱时出错率高等。因此如果要对这个系统进行改进，

提高使用的便利程度将是另一个重点。主要的工作方向可能有：

1. 可以考虑以音素而不是音节为单位来合成声乐。这将大大减小工作量，也将大大增加可以合成的语言种类。当然，这对辅音与元音连接的平滑性提出了较高的要求。
2. 编写可视化的乐谱编辑器，以克服 txt 格式由于要求词曲对齐带来的限制。

7. 参考文献

[1] Murtaza Bulut, Shrikanth Narayanan & Lewis Johnson, "Synthesizing Expressive Speech Overview: Challenges, and Open Questions", pp. 175-201 in "Text to Speech Synthesis: New Paradigms and Advances", Shrikanth Narayanan & Abeer Alwan, 2004

[2] M. Mohan Sondhi & Daniel J. Sinder, "Articulatory Modeling: A Role in Concatenative Text to Speech Synthesis", pp. 63-87 in "Text to Speech Synthesis: New Paradigms and Advances", Shrikanth Narayanan & Abeer Alwan, 2004

[3] Ellen Eide, Raimo Bakis, Wael Hamza & John F. Pitrelli, "Toward Expressive Synthetic Speech", pp. 219-250 in "Text to Speech Synthesis: New Paradigms and Advances", Shrikanth Narayanan & Abeer Alwan, 2004

[4] 《数字声音处理》，[日]古井贞熙 著，朱家新 张国海 易武秀 译，东海大学出版会，1985

[5] 《WAV 文件格式》(http://www.kk.iij4u.or.jp/~kondo/wave)，Kondo Masayoshi

[6] 《语音处理》，[美]Thomas W. Parsons 著，文成义 常国岑 王化周 赖金福 译，McGraw-Hill Book Company, 1986, 国防工业出版社，1990

[7] MBROLA 项目的网站，http://tcts.fpms.ac.be/synthesis/mbrola.html

8. 附录

8.1 声乐合成实验的源程序和乐谱示例

本实验的源程序和乐谱示例与论文一同放在了压缩包 VocalSynth.rar 中。压缩包的目录结构如下：

- Program 目录：包含数据库建立程序和声乐最终合成程序的源代码 prepare.cpp 和 synth.cpp。程序开头的宏定义中设置了两个程序的工作目录，请注意修改。
- Database 目录：存放数据库。对于每个歌手，有一个以歌手名为名的文件夹，其下又有 input、output 两个目录。input 目录下存放原始语音材料（*.wav）和音节表（*.txt），并有一个 list_in.txt 文件，描述这些文件的信息。list_in.txt 的格式为：若干行，每行四项，分别为：音节表文件名，原始语音材料文件名，最低频率，最高频率。output 目录下存放由程序生成的数据库。由于附件大小的限制，原始语音材料没能放进压缩包。
- Music 目录：包含 accom, score, synth 三个子目录，分别存放伴奏、乐谱、合成声乐文件。由于附件大小的限制，只提供了乐谱示例。
- Thesis 目录：包含本篇论文。

8.2 对“信息科学技术概论”课的感想

上了一个学期的信息科技概论课，我对它有着多方面的感受。

这门课的内容范围很广，从人工智能到网络安全，既有技术性比较高的课题，也有与社会联系比较紧密的领域，让我们大开眼界。另外，姚期智教授的英文授课也让我听得很爽，也给了曾经搞过信息学竞赛的我一定的启发。

然而，我觉得这门课有一个巨大的缺陷，就是起点太高，离大多数同学的基础太远。许多东西只是简单地提一下都取得了哪些成果，而没有详细地甚至粗略地介绍这些成果是什么，有什么用途，听起来就像天书，以致于好多同学在课堂上做作业、睡觉甚至翘课。加入一两堂英文课是好的，但是应该考虑到同学们的知识基础，我的专业英语还算好一些，大部分同学不知道一些术语的英文说法，以致听不懂。因此，我建议这门课能够贴近学生的基础，在英语授课时，能够先提供一些术语词汇。

由于大多数的课只是让我们饱了眼福，看了热闹，并没有给我们实质性的帮助，所以寒假中的论文对许多同学来说都很困难。固然，我参阅了一定量的文献，但从网络上浩如烟海的资料中寻找贴切的资料十分困难，且由于知识基础的薄弱，大部分都看不懂，以致于我经历十多天搞的这样一个庞大的实验，用到的大部分知识都是以前的积累，处于一种严重的“吃老本”的境地。因此，我希望这门课能够真正地启发着我们寻找研究课题，并对查阅文献的方法做出一定的指导。

网络学堂上活跃的讨论，算是这门课比较成功的地方。如果这门课能够弥补上述缺陷，使同学们真正学有所得，我相信网络学堂上的讨论会更充实。