Spectral modelling and timbre hybridisation programs for computer music*

JAMES BEAUCHAMP† and ANDREW HORNER‡

† School of Music, University of Illinois at Urbana-Champaign, 2136 Music Building, 1114 W. Nevada, Urbana, IL 61801, USA
E-mail: j-beauch@uiuc.edu
‡ Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
E-mail: horner@cs.ust.hk

Software packages developed for visualising time-varying timbres, finding parameter settings for modelling acoustic instrument tones, and for synthesising timbres are described. The *sndan* package, written in C for Unix machines, provides spectrum analysis, pitch tracking, spectrum display, parameter modification, and additive synthesis. Other programs that estimate the best possible parameters for recreating acoustic sounds using techniques such as multiple FM and wavetable synthesis and extended nonlinear/filter synthesis are outlined. Hybrid instruments created by fashioning sounds out of the recombined spectral characteristics of two or more very different tones are also presented.

1. INTRODUCTION

We have developed a package of programs for sound analysis and synthesis, designing sound synthesis models, and creating hybrid timbres. The programs help visualise the time-varying timbre of sounds, discover good synthesis parameter settings for matching particular instruments, and assist in the synthesis of timbres.

The basic package, called *sndan*, written in C for Unix, is used for time-varying spectrum analysis of sounds (both fixed band and tracking), pitch tracking, parameter visualisation and modification, and additive synthesis. We have also written a *NeXTStep* version of the package, called *AnView*. Both *sndan* and *AnView* can be obtained by ftp from our site on the Internet.[1]

The timbre modification possibility can be interesting for music perceptual scientists who wish to determine the effects of various parameter modifications and the relative importances of various parameters within particular sounds. Sound modifications can also be used directly in computer music compositions. The display capability of *sndan* is useful for visualising timbral quality and for documentation; hopefully, the several graphs generated by *sndan* in this paper will illustrate this claim.

*This work was supported by the Research Grant Council's Project HKUST729/96E.

[1] http://cmp-rs.music.uiuc.edu/cmp/software/sndan.html

Based on *sndan*, we have experimented with a variety of synthesis models, such as FM and wavetable synthesis, which can mimic acoustic sounds. We have used a combination of techniques for determining parameters that enable model outputs to closely match the time-varying spectra of original acoustic sounds. These include a genetic algorithm, principal component analysis, and least-squares minimisation. We have found that for most musical instruments only three to five FM carriers or wavetables are required for a good synthesis match (Horner, Beauchamp and Haken 1993a, b). We can even choose wavetables which work over a wide range of pitch (Horner, Beauchamp and Haken 1992). We have also determined parameters for generalised nonlinear/filter synthesis (Beauchamp and Horner 1992) and have achieved good matches to original sounds using two or three nonlinear modules.

FM, wavetable, and nonlinear/filter synthesis models are lightweight users of computer time. However, they have problems with parameter sensitivity, making it difficult to control them as synthesis instruments. A new method, *spectral envelope synthesis* (Beauchamp and Horner 1995), shows much promise in this regard, since it is able to match the average spectral characteristic of a particular instrument over its pitch and dynamic range while matching the spectral centroid (brightness) vs time envelope of a particular tone.

In addition, we have also explored two new techniques for creating 'hybrid instruments'. Both methods manipulate spectral information produced by *sndan*. One method scales the harmonic amplitudes of a sound to achieve the principal spectrum of another. The method retains the envelope structure of the first sound, and its quality usually prevails. The second method we call *timbre breeding*, a natural selection-based procedure, whose goal is to generate interesting sounds. Starting with two original source sounds (e.g. a trumpet and a rubbed glass), it iteratively applies filtering and time warping operations to arrive at a hybrid which combines aspects of both sounds.

Organised Sound **2**(3): 253–8 © 1997 Cambridge University Press. Printed in the United Kingdom.

https://doi.org/10.1017/S135577189800908X Published online by Cambridge University Press

2. USE OF *SNDAN*: SOME EXAMPLES

The principal functions that are available within the *sndan* package are shown in figure 1. Two types of time-varying analysis are available: pitch-synchronous harmonic filter bank (*pvan*) and non-pitch-synchronous spectrum peak tracking (*mqan*). The first results in a series of harmonics relative to a chosen fixed fundamental with variable amplitudes and frequencies. The second gives a series of 'tracks', each having a frequency and amplitude, but without regard to whether the frequencies are experiencing a lot of variation or whether they are harmonic or inharmonic. However, if they are harmonic, pitch detection (using *fcheck*) and harmonic separation (*harmformat*) can be performed, and a harmonic analysis file produced. Once a harmonic analysis file is produced (by either *pvan* or the *mqan–fcheck–harmformat* combination), *monan* is the main program used for parameter display, parameter modification, and additive synthesis. *addsyn*, *addsyn1* and *addsyn2* are standalone programs for additive synthesis that allow changing the pitch and duration of the sounds independently. Inharmonic analysis files can be graphed with *mqdistr* (track display) and *mqdispp* (3D spectrum plot) and synthesised with *mqsyn2* (additive synthesis with time scaling), but these programs do not yet have the versatility of *monan*.

Figure 2 illustrates a series of *monan* operations applied to the amplitude envelopes of the harmonics of a sung vocal 'ah' sound. The envelopes are first smoothed using a 10 Hz cut-off low-pass filter. When resynthesised, this removes much of the noise or hoarseness in the voice. The second operation tests the degree to which the amplitude envelopes must be different, since all envelopes are set to the same rms envelope shape with only amplitude multipliers to differentiate them. The average spectrum remains the same, however. The main difference is heard during the attack and decay of the sound.

Figure 3 shows a similar series of operations on frequency data of the same sound. The data is plotted on a normalised scale according to a range of −0.03 to +0.03 times the analysis frequency (130 Hz). The data is first smoothed by a 10 Hz low-pass filter. This removes the remaining noise in the voice, but there is still some natural pitch fluctuation. However, the next step, which zeros all frequency fluctuations, reduces the voice to 'Robby the robot'.

The previous sound example was easily handled by *pvan* because its frequency fluctuations were quite small. However, when the frequency fluctuations are large, as in the case of an operatic tenor voice with vibrato, a fixed filter bank gives grossly inaccurate results. A spectrum peak tracking analyser works much better. Figure 4(a) shows the vocal pitch of a tenor voice (vowel: 'ah') detected by the *mqan–fcheck–harmformat* combination, on a scale of ±6%, or one semitone. The 3D spectrum for this sound is shown in figure 4(b). Why is it that some harmonic amplitudes rise and fall so much, with some out of phase with the others, whereas other harmonic amplitudes (notably harmonics 1, 2 and 3) show little variation? The answer can be seen by looking at figure 4(c), which shows traces of amplitude vs frequency curves for the first forty-eight harmonics. These traces, for the most part, overlap to form a vocal 'filter characteristic', with the first two formant peaks at about 800 and 1,000 Hz, and the third 'singer's' formant at 2,600 Hz.

These are just some examples of operations that can be done with *monan* in *sndan*. *monan* currently includes about thirty commands to perform various display, modification, and file I/O operations. Since *monan*, like other *sndan* programs, is written in C in a modular fashion, it is easy for anyone versed in C to install new custom commands.

3. SYNTHESIS MODEL DEVELOPMENT

Synthesis models are objects which can be used to produce particular types of sounds. One of our goals is to utilise simple synthesis models for emulation of

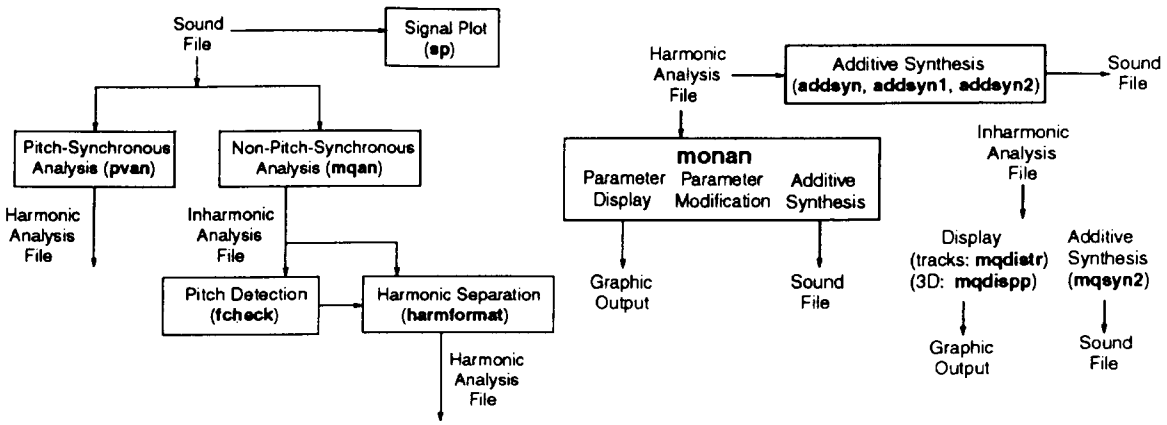


Figure 1. *sndan* program organisation.

https://doi.org/10.1017/S135577189800908X Published online by Cambridge University Press

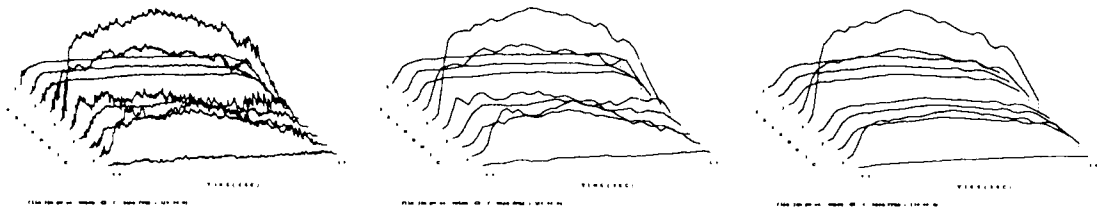


Figure 2. Operations with *monan* on harmonic amplitude envelopes of the sung vowel 'ah'.

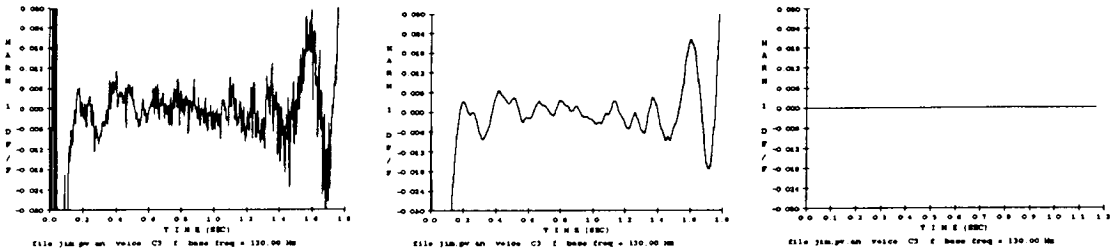


Figure 3. Operations with *monan* on harmonic frequency envelopes of the sung vowel 'ah'.

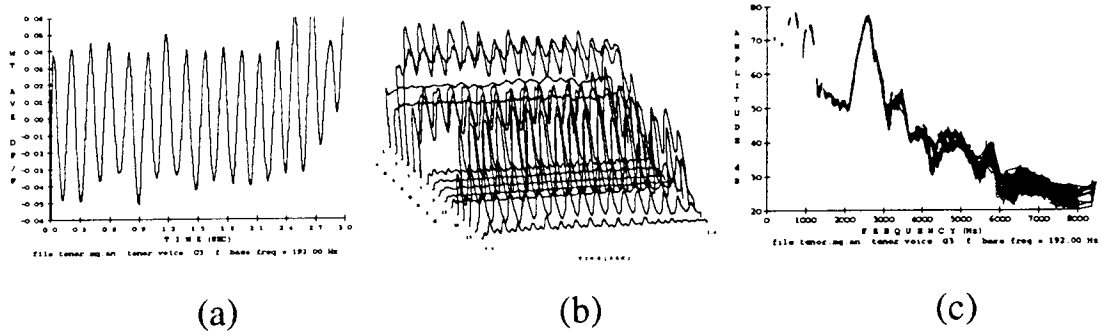


Figure 4. Tenor vocal tone parameters: (a) pitch vs time, (b) time-varying harmonic spectrum, and (c) amplitude vs frequency traces.

acoustical instrument sounds, a long-standing problem in computer music. We started by trying traditional methods, such as frequency modulation, wavetable interpolation, and nonlinear/filter synthesis. These have been amply described in other articles, so we will just give an overview here. Our first project was to find an FM model (Horner, Beauchamp and Haken 1993a) whose time-varying output spectrum closely matches that of a particular acoustic instrument sound. Time-variant analysis and synthesis were accomplished using *sndan* routines.

We chose an FM model having a single modulator and N carriers which were added together with time-varying weights. The problem was to determine the best carrier-to-FM ratios (fixed integer values), FM indices, and carrier weights (amplitudes) for matching the output spectrum to that of the original sound. Since the FM problem was generally intractable with traditional optimisation techniques, we used a genetic algorithm to determine the ratios and indices. We soon found that keeping the indices fixed greatly simplified the problem without sacrificing quality. Once the ratios and indices were determined, the time-varying carrier amplitudes were found using a straightforward least-squares method. The result was that good matches to original sounds could be achieved using three to five carriers for most harmonic spectrum instruments, such as trumpet, voice and guitar.

At this point, we noted that since the FM indices were fixed, the carriers could be represented as fixed wavetables. But it was obvious that we need not restrict ourselves to FM-generated wavetables. Therefore, we decided to generalise the problem to that of finding the N best arbitrary wavetables for simulating a particular sound (Horner, Beauchamp and Haken 1993b). We tried two different strategies: (i) choosing waveforms from the sound itself, and (ii) choosing waveforms based on principal component analysis of the sound's time-varying spectrum. We found that method (i) yielded the best results, especially when less than four wavetables were used and that, again, three to five wavetables were sufficient for imitating most sounds. We even found that by selecting wavetables from various pitches of a

https://doi.org/10.1017/S135577189800908X Published online by Cambridge University Press

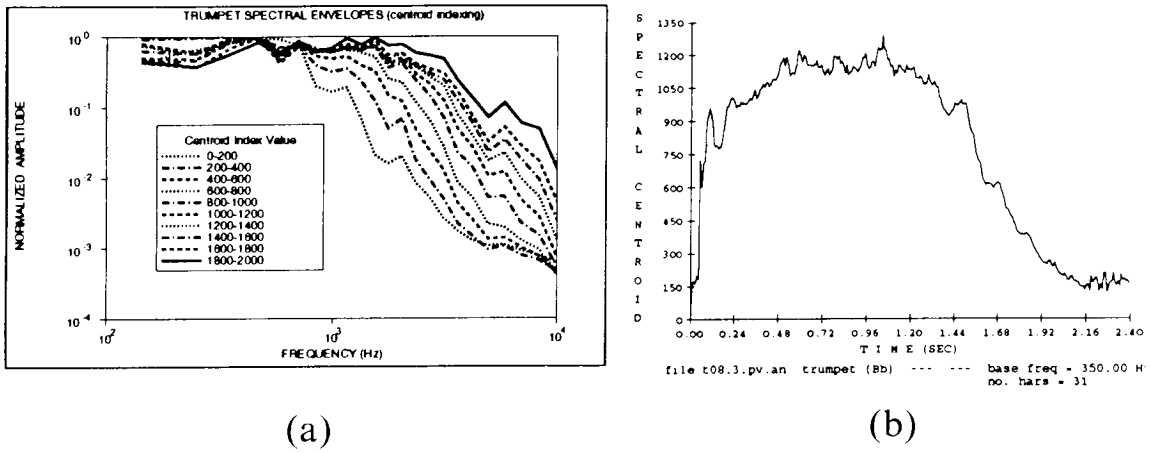


Figure 5. Trumpet synthesis parameters: (a) spectral envelope family indexed by spectral centroid, and (b) spectral centroid vs time.

given instrument, good results could be obtained over substantial pitch ranges (Horner, Beauchamp and Haken 1992) using a few wavetables.

A nonlinear/filter model was also tried (Beauchamp and Horner 1992) using N nonlinear distortion modules. Some advantages of this source/filter method were: (i) only two or three nonlinear modules were needed in many cases, (ii) spectral centroid (or 'brightness') of the sound could be easily varied, and (iii) the built-in high-pass filter provided a natural way for spectrum to vary with pitch.

These methods failed to satisfy a basic goal, however, spurring development of a new method. The goal is that a few simple, easily understandable control functions be sufficient to vary the timbral quality of an instrument in an appropriate way for any pitch and dynamic. Our hypothesis for development of this model is that spectral data can be indexed according to some spectral measure independent of amplitude and pitch. We have chosen the spectral centroid as this measure. By first dividing all spectra generated by a particular instrument into different centroid zones and averaging these spectra, we have arrived at a family of spectral envelopes. For example, a family for a trumpet is shown in figure 5(a). A sound is generated by first sweeping through the spectral envelopes while sampling them at the harmonic frequencies of the sound. The sweep occurs according to the desired spectral centroid envelope (a typical one is shown in figure 5(b)), which can be thought of as analogous to an FM index envelope.

Then, the desired frequency and amplitude envelopes are applied. Synthesis can proceed by additive synthesis or more efficiently by using interpolated wavetables in a mathematically equivalent fashion. When the spectral centroid, frequency and amplitude envelopes are taken from original tones, the resulting synthetic tones closely resemble these originals.

We have designed a Music 4C trumpet instrument (available by ftp)[2] using the spectral envelope synthesis algorithm and arrived at the following parameters:

(1) pitch (octave and pitch-within-octave),
(2) amplitude factor (0 to 10),
(3) brightness factor (0 to 10),
(4) bite (0 to 2) (this controls extra brightness just at the tone's beginning),
(5) envelope shape,
(6) attack time (typically 0.5 s),
(7) decay time (typically 1.5 s),
(8) fraction of random amplitude (typically 0.05),
(9) fraction of random frequency (typically 0.003), and
(10) cut-off frequency for random variations of amplitude and frequency (typically 6 Hz).

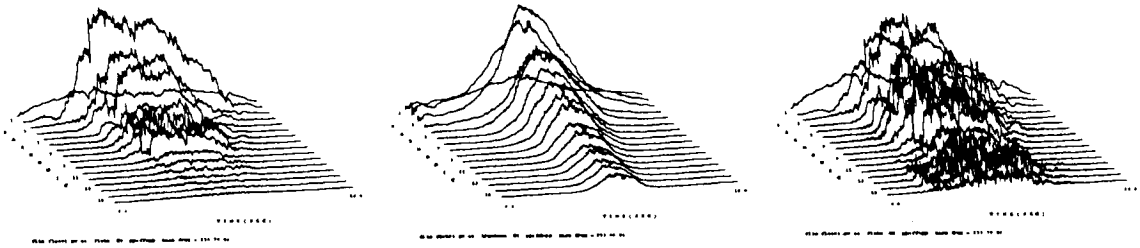
Smooth envelope shapes were used, but for naturalness we added a certain amount of randomness to both frequency and amplitude. We found that amplitude jitter was more critical than frequency jitter. Although the resulting sounds were not perfectly natural-sounding, they were a lot more so than sounds without any random variations.

4. HYBRID SOUNDS

The first type of hybrid sound was designed to demonstrate the relative importance of spectral evolutions compared to steady-state spectra. First the steady-state spectrum of sound B is measured. Then the harmonic amplitudes of sound A are multiplied by factors in order that at some instant within sound A's duration it achieves the spectrum of sound B. Figure 6 shows an example using flute and trombone tones.

[2] http://cmp-rs.music.uiuc.edu/cmp/software/m4c.html

https://doi.org/10.1017/S135577189800908X Published online by Cambridge University Press



Sound A – flute C4 **Sound B – trombone C4** **Flute equalized to trombone**

Figure 6. Spectral of original flute and trombone tones, and synthesised flute equalised to the trombone.

This is tantamount to putting sound A through a very precise graphic equaliser – so precise that it can adjust the level of each harmonic differently, in an attempt to achieve the quality of sound B. The question is: does the resulting sound resemble A or B more closely? Most people will say A, which supports the theory that spectral evolution is most important for timbral classification. At any rate, this is an interesting method for producing interesting and ambiguous timbres, that may have some possibilities for composition.

A more complex method for producing hybrid sounds is one we call 'timbre breeding' (Horner, Beauchamp and Packard 1993). Initially, a source sound is selected for breeding. The source tone is modified by a series of random filtering and time warping operations. The resulting offspring sound is then compared to its parent. The user decides which sound survives and becomes the source for the next generation.

The process continues until the user is satisfied with the evolved sound. Alternatively, interesting sounds may be saved along the way which together form a timbral evolution of tones. The number of filtering operations applied to the tone is controlled by the user, and can be changed at each generation if desired. The mutation rate for time warping works similarly.

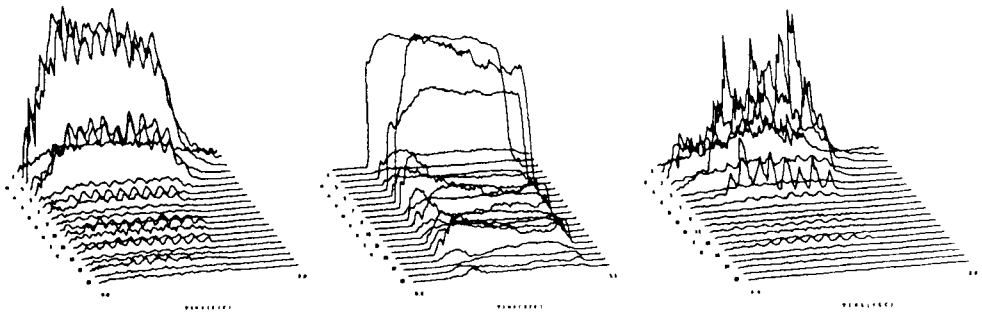
Each filtering operation is only applied to a portion of the source tone. Four time points are randomly chosen which determine when the filter is active. The filter is activated at the earliest of these time points. However, the filter is not fully active until the second time point. The deactivation of the filter follows a similar pattern. We use a normalised short-time spectral snapshot of another tone as the filter function. A different spectral snapshot is used for each filtering operation. The snapshots are chosen from a tone (or tones) selected by the user to mate with the source tone.

As with filtering, each time warping operation only affects a portion of the sound. Two time points are randomly chosen which delimit the time warping period. A compression factor is also randomly selected. It controls the extent of the time expansion/compression.

The final offspring is frequently a sound which is radically different from either of the two source sounds but, at the same time, retains some elements of these sounds. Figure 7 shows an example based on cello and bass clarinet original files.

5. SUMMARY

We have briefly described the *sndan* package for analysis/synthesis and some of our recent and current research in synthesis models. We believe that use of such software and synthesis methods will prove beneficial for music research and composition in the future.



Cello **Bass clarinet** **Timbre Bleeding Offspring**

Figure 7. Spectra of original cello and bass clarinet tones, and timbre breeding offspring.

https://doi.org/10.1017/S135577189800908X Published online by Cambridge University Press

REFERENCES

Beauchamp, J., and Horner, A. 1992. Extended nonlinear waveshaping analysis/synthesis techniques. *Proc. of the 1992 Int. Computer Music Conf.*, pp. 2–5. San Jose, CA: ICMA.

Beauchamp, J., and Horner, A. 1995. *Wavetable Interpolation Synthesis Based on Time-Variant Spectral Analysis of Musical Sounds*. Audio Engineering Society Preprint No. 3960.

Horner, A., Beauchamp, J., and Haken, L. 1992. Wavetable and FM matching synthesis of musical instrument tones. *Proc. of the 1992 Int. Computer Music Conf.*, pp. 18–21. San Jose, CA: ICMA.

Horner, A., Beauchamp, J., and Haken, L. 1993a. Machine tongues XVI: genetic algorithms and their application to FM matching synthesis. *Computer Music Journal* **17**(4): 17–29.

Horner, A., Beauchamp, J., and Haken, L. 1993b. Methods for matching multiple wavetable synthesis of musical instrument tones. *Journal of the Audio Engineering Society* **41**(5): 336–56.

Horner, A., Beauchamp, J., and Packard, N. 1993. Timbre breeding. *Proc. of the 1993 Int. Computer Music Conf.*, pp. 396–8. Tokyo: ICMA.

APPENDIX

sndan sound examples

1. Original voice tone
2. Original voice resynthesised with 63 harmonics
3. Original voice resynthesised with 32 harmonics
4. Original voice with amplitude and frequency variations smoothed
5. Original voice with amplitudes replaced by rms equivalents
6. Original voice with frequencies set to fixed harmonics
7. Original voice again
8. Trombone tone
9. Voice envelopes scaled for trombone spectrum
10. Trombone envelopes scaled for voice spectrum

Timbre breeding sound examples

1. Cello and bass clarinet offspring #1
2. Cello and bass clarinet offspring #2
3. Cello and bass clarinet offspring #3
4. Rubbed glass tone
5. Original trumpet chords
6. Rubbed glass and trumpet chords offspring #1
7. Rubbed glass and trumpet chords offspring #2
8. Rubbed glass and trumpet chords offspring #3
9. Chinese pipa tone
10. Chinese pipa and rubbed glass offspring

https://doi.org/10.1017/S135577189800908X Published online by Cambridge University Press