FOUR REALIZATIONS OF FOREST CANOPY RETRANSLOCATION DATA

Arun Chandra

COM 301
The Evergreen State College
Olympia, WA 98505
arunc@evergreen.edu

Nalini Nadkarni

Lab II
The Evergreen State College
Olympia, WA 98505
nadkarnn@evergreen.edu

ABSTRACT

This is a report on four realizations of data from the Forest Canopy Project, directed by Dr. Nalini Nadkarni at The Evergreen State College. Data collected over 35 months in the forests of Costa Rica on the retranslocation of chemicals from four tree species were realized into four-voiced sounds in a variety of ways: band-limited white noise, glissandi using "chords" of varying frequency content, articulated "chords" changing in density of attacks, and synthesized speech.

Each way had its benefits and drawbacks, and this paper will attempt to report those aspects. Custom-built software (utilizing *python* and *C*) was used to translate the data into CD-quality soundfiles (16-bit, stereo, 44100 sampling rate).

1. INTRODUCTION

Over a period of almost three years, Dr. Nalini Nadkarni and her assistants collected data on how much nitrogen, calcium, and phosphorus remained in leaves that fell from four tree species in Costa Rica: *Ficus*, *Meliosma*, *Ocotea*, and *Conostegia*. Exploring alternative means of presenting data to the lay public, Dr. Nadkarni asked me if I would be interested in taking the data, and realizing it in sound. She and I were curious as to what this realization might tell about the data, and whether such a realization might make the data "comprehensible" (in some fashion) to the lay public.

We developed four methods of realizing the data. All four were programmed using a combination of *python* and *C*. In *python* were coded the procedures for translating the original data files into a format usable for the sound-generation routines, coded in *C* for speed. The output was a stereo sound-file of 16-bit samples at a sampling rate of 44100 samples-per-second.

2. DATA MAPPING TO FREQUENCY, DURATION, VOLUME, AND STEREO LOCATION

An example of input data is given in *Figure 1*. The four lines of data represent the four recorded species *Ficus*, *Meliosma*, *Ocotea*, and *Conostegia*. These files show "...how much of the nutrient has been pulled out as a proportion of the total" over 35 months. There were three sets of datafiles, for calcium, phosphorus, and nitrogen.

The *python* program took these datafiles as input, and translated them so that the *y* values (the percentage of a chemical in the species) were logarithmically mapped into *non-overlapping* frequency bands. For example, species 1: 100–200Hz, species 2: 200–400Hz, species 3: 400–800Hz, species 4: 800–1600Hz.

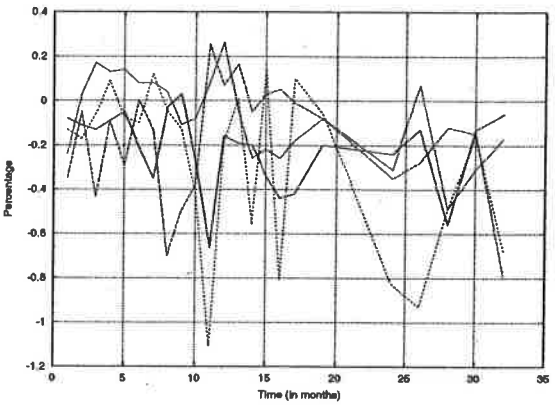

Figure 1: *An example of the input data (Calcium)*

In our first attempt, we mapped the input data into *overlapping* frequency bands. We had made the assumption that we needed to preserve the relative data values of the four species, and scale them equally over the full frequency band. However, this overlap made the differences between the four species very difficult to detect. Their differences were more easily discernible when *non*-overlapping bands were used.

The *x* values (indicating the month in which the data were recorded) were mapped into a user-scaleable *time*. Figure 2 shows what the translation looks like for the *Calcium* input data (Figure 1) when the *time* is chosen to be 30 seconds.

The overall frequency range for the four species data was easily changeable by us, and the total duration as well. We found the most useful durations to be between 30–120 seconds.

In addition to mapping the *y* values into frequency, they were also mapped into decibels, so that as the amount of a chemical in a species rose, the volume of the sound grew. The decibel range could have been variable, but was generally left at 76–92dB.

The "voice" for each species was given a distinct location in the stereo field: left, right, left-of-center, and right-of-center.

3. SOUND GENERATION

Four sound generation routines (coded in *C*) were used. (Sound-generation routines coded in *python* were tried, and rejected for being too slow for the quality of result we wanted.)

1. Band-limited white-noise.
2. Glissandi consisting of "chords."

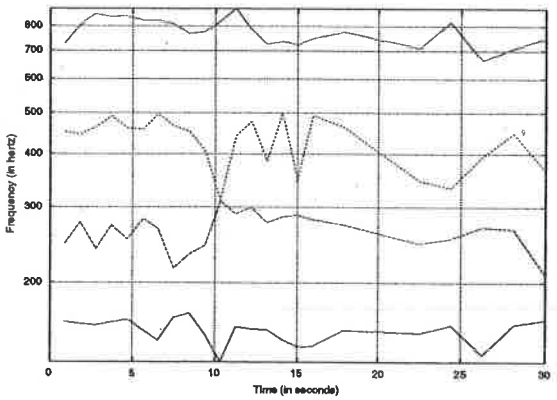

Figure 2: *Rescaled Calcium data*

3. Individual "chords" that changed their density proportional to the magnitude of the y value.
4. Speech synthesis in which the density of "words" was proportional to the y value.

The data points (the x, y pairs) where then used as starting and ending values for the sound parameters.

3.1. Band-limited white-noise

A uniform random-number generator was used to create 16-bits of white noise. These values were passed through a band-pass Butterworth filter, whose passing width was set to 5 cents. (Wider widths were tried, but 5 cents allowed for the greatest differentiation between the 4 "voices." Wider widths had the consequence of "diffusing" the presence of each "voice.")

Each "voice" then used the initial data point as a starting value, changed to the next data point, and continued until the end.

The success of this method of realization was undeniably aesthetic: the stereo result sounded like "the wind in the trees," and was immediately enticing to all its listeners.

The problems of this method lay in differentiating the four distinct species (the "voices"). It was only with difficulty that three voices could be discerned—insufficient for what we wanted.

3.2. Glissandi consisting of "chords"

Given the difficulties of discerning four "voices" when using band-limited white noise, we decided that the cause of this difficulty was the timbral similarity between the four "voices": each was white-noise, band-limited to 5 cents.

We then tried using glissandi consisting of distinct "chords" at each data point, and these "chords" would then move from the one data point to the next.

A "chord" was defined as being a sine tone with up to N overtones, randomly chosen for each data point. The fundamental frequency was determined by the original data point. The frequency of each overtone was randomly chosen from a set of diatonic intervals (the set of intervals available on a piano, using both the white- and black-keys).

The amplitude of each overtone was set at $\frac{1}{n}$, where n was the overtone number. (This is identical in structure to the amplitude of overtones in a sawtooth wave.)

A "glissando" was defined as the logarithmic change of frequency of a fundamental and its overtones to a destination fundamental and its overtones.[2] (When sequential points differed in the number of overtones present, the extra values used the fundamental as a starting or ending point.)

This technique allowed a trained ear to discern all four "voices", but untrained ears still had a difficult time discerning them. Thus, this technique was also deemed insufficient for what we were hoping.

3.3. Changing density of individual "chords"

As reported in [1], the "Geiger-counter" is an example of the most successful example of sonification. We decided to try this method on our data in the following way: the data were analyzed so that the greatest magnitude of a "voice" would be the location of the greatest number and greatest dynamic volume of "chords".

The duration of a "chord" was fixed to a range of 0.075 to 0.050 seconds. The silence that followed a chord was allowed to vary between 0.75 seconds and the current duration of a "chord".

Although the duration of a chord was randomly chosen within the limits above, the duration of the following silence depended on the current y value of the data.

The frequency content of a "chord" was determined in the same way as described above for the "glissandi."

Each "chord" was given a random ADSR envelope, and the "attack" portion was given an added random noise. Since the duration of the chords was quite short, the addition of the random noise served to give each attack a "crispness," as if it were being created by a percussive instrument (guitar, piano, marimba, etc.).

This technique was the most successful of the four, since the four data lines were clearly perceivable by the untrained ear.

3.4. Speech synthesis

The technique of "speech synthesis" was in some ways the least useful, while at the same time being the most humorous: it sounds like four people chattering at the same time, and occasionally getting excited. The freely-available speech synthesizer MBROLA [3] was used, along with one of its phoneme databases.

The data analysis was the same as that for the "chords," mentioned above: the frequency and amplitude were directly proportional to the y values in the data. The duration, too, was set in a similar fashion.

Each "word" of the speech consisted of a duration, frequency, an initial consonant, a vowel, and a final consonant. The two consonants and the vowel were randomly chosen from a set of available choices. The frequency was determined by the y variable, as well as the dynamic. The density of "words" was determined in a similar way to the "chords" above.

Each "voice" was spread out over the stereo field, as in the above attempts.

The resulting sounds, although enormously amusing to hear, were the least helpful for data analysis. A potential experiment, as yet to be tried, is to place the four "voices" in four separate channels, through four separate speakers, and compare the results to the current stereo version.

4. REFERENCES

[1] Kramer, G. *et al.* "Sonification Report: Status of the Field and Research Agenda." http://www.icad.org/websiteV2.0/References/nsf.html, 1997.

[2] Moore, F. Richard, *Elements of Computer Music*, Prentice Hall, 1990.

[3] T. Dutoit, V. Pagel, N. Pierret, F. Bataille, O. Van der Vrecken *The MBROLA Project: Towards a Set of High-Quality Speech Synthesizers Free of Use for Non-Commercial Purposes* Proc. ICSLP'96, Philadelphia, vol. 3, pp. 1393-1396.