Timothy Opie
Queensland University of Technology
Music and Sound
Kelvin Grove, 4059
Australia
timopie@fastmail.fm

Fundamental Frequency as an Attribute of Eco-Structuralism

Abstract

Eco-structuralism is a compositional technique that captures and transforms audio structures derived from the environment. The transformations are done in accordance within prescribed constraints of eco-structuralism to preserve ecological information relating to the sound. This paper focuses on the fundamental frequency aspects of this technique.

Introduction

Eco-structuralism has roots going as far back as antiquity. Mimetic influences from Aristotle have had a far-reaching effect on music (Dahlhaus 1982). This relationship has changed as composers have moved towards more abstract methods of thinking about music, and indeed perceiving music (Kivy 2002). The change in these ideas of music and its role in society paved the way for compositional processes such as *Serialism, Musique Concrète* and *Soundscape Composition*.

This paper will briefly explore the history of environmentally based compositional methods and how they relate to eco-structuralism. It will then take a critical look at a single sound attribute and explore how it can be used within eco-structuralism. The sound attribute to be focused on is fundamental frequency. Some of the problems and solutions associated with analysis and recreation of the fundamental frequency will also be examined.

Environmentally Based Composition

The influence of nature and environmental patterns in music composition has a long history. For example, Vivaldi's *Le quattro stagioni* (*The Four Seasons*) published in 1725 is a highly recognised musical work that demonstrates a clear mimetic influence. In fact the underlying philosophy and practice of music was that it should imitate nature, and praise the creator of nature. During the Renaissance, as the influence of Aristotle was being slowly stripped away and even more so as people such as Descartes began questioning existence and creation, the idea of imitation became too limiting (Dalhaus 1982). Music also followed this ideology through a number of musical genres. The pastoral music by composers such as Beethoven was more about the creation of life and nature, than about imitation of nature. Ludwig van Beethoven's Symphony No.6, Op.68 *Pastoral*, written in 1808 is a major example of this genre. Musical realism was also a significant genre that no longer praised or imitated nature, but attempted to depict life in all its shades of beauty and ugliness (Dalhaus 1982). It became a representational model of life, abstracted from the original. Further abstraction from nature led the way to 12 tone music and serialism (Schoenberg 1975). In the 1940s *Musique Concrète* took abstraction to a different plane. Instead of creating sound with instruments, Pierre Schaeffer recorded sounds, and used them in an abstract setting (Field 2000, Hodgkinson 2001). Jean Claude Risset's *Sud* created in 1985 is a piece that directly uses the sounds of birds, oceans and insects alongside the sounds of wood and metal chimes, piano and computer generated sounds. This piece hybridizes some sounds by imparting the dynamic characteristics of one sound to another.

In the 1970s R. Murray Schafer also began using recorded sounds from nature, but his goal was to contextualize them back into nature. Degeneration of the audio landscape, sound pollution, and sound intrusions in general have only in recent decades been acknowledged as a threat to our environment (Schafer 1994). Schafer began the first studies into this area, and founded the World Soundscape Project (WSP) (Truax 1999). The WSP developed categories for attributes of the soundscape and made and analysed recordings of soundscapes in Canada, making assessment on keynotes, sound events, and soundmarks. Barry Truax was among those on his research team.

Damian Keller worked with Truax in an area now called eco-composition, which has added the complexities of ecology into the compositional process with soundscapes (Keller and Capasso 2006). The ecological framework captures not only the sound event, but also the personal history of the sound event as perceived by the composer, and later the listener (Gibson 1966). It follows the ideas of mimesis as extended from Aristotle by Paul Ricoeur. For example, when one listens to the sound of a river one is encouraged to reflect on the significance of a running river in their memory, and impose those ideas on the composition (Keller 2000).

Eco-structuralism formalises a compositional method for creating musical works with soundscapes and eco-compositions. It sets up rules for creating musical structures and articulates the ways in which these structures may be manipulated in a musical fashion. The rules of eco-structuralism are based on a form of serialism in which sonic structures must retain their form (Opie 2006). The attribute study that has been conducted, alongside the creation of eco-structuralism, is a way of investigating the structures and looking at ways in which structures can be formed or extracted from audio events, and how they can be manipulated in a musical context. By keeping the musical structure intact it is supposed that the ecological information will also re-

main intact, despite a number of musical changes to the structure.

Fundamental Frequency

The study of fundamental frequency is part of a larger attribute study I have been conducting, which aims to catalogue and make use of the various audio attributes of microsized audio. Microsized audio meaning a range of sound from 20ms to 1sec. Fundamental frequency was the second attribute studied in the attribute study; the first was amplitude (Opie, 2005).

The fundamental frequency attribute study is broken into a number of methods:

- Extraction
- Synthesis
- Scaling
- Substitution
- Generation

Within each of these methods one or more programs have been written that utilise the method. Extraction is the method of gathering data from an audio file. Synthesis is the method of recreating an identical audio file using the data gathered from the extraction method. Scaling is the method of performing a number of different scaling techniques, such as time scaling, pitch scaling, amplitude scaling, etc. Substitution is the method of substituting one audio attribute for another. Generation is the method of generating unique audio files using a structure algorithm based on an analysis of the data extracted.

Extraction

Procedure:

The program *ExtractFrequency.java* is designed to create a data file containing the primary analysis information. The information is gathered using a data reduction process that assesses and records the number of samples per cycle throughout the audio file. Firstly the program traverses the audio sample looking for zero crossings, that is, the point where a waveform crosses from a negative to a positive value, or vice versa. It then evaluates the number of samples contained between a pair of zero crossings. This value is recorded in a data file. The data file itself does not contain the actual frequency value of the fundamental frequency. It contains the more functional value of the samples per cycle value. This method has been chosen because it is more functional and much quicker to use in computation. The samples per cycle method also affords simpler scaling techniques.

If the composer wishes to view the actual frequency data, as a frequency, they can be evaluated using the *spc2frequency.java* program which uses the two algorithms to generate both the instantaneous fundamental frequency, and the average fundamental frequency.

The instantaneous fundamental frequency of the single cycle is evaluated using the equation:

```
value(1)/samplerate
```

The average fundamental frequency is evaluated in integer (s) seconds using the equation:

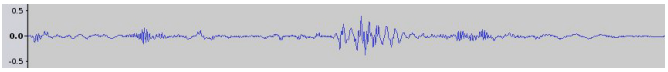
```
When (value(1) + … + value(n) ==
samplerate*s)

(value(1) + … + value(n))/n =
average frequency
```

The algorithm used in *extractfrequency.java* to detect zero crossings is listed below:

```
if(((infoBuffer[counter]>0f) &&
    (infoBuffer[counter+1]<0f))||
   ((infoBuffer[counter]<0f) &&
    (infoBuffer[counter+1]>0f)))
```

where counter is a place keeper in the audio file.

There is also a second counter that counts the number of samples since the last zero crossing, which is the value recorded in the data file.



The above graphical representation of a babbling brook will create a data file similar to the following:

```
Extracted Fundamental Frequency Data
VER: 1.0, By Timothy Opie
5,7,6,7,5,16,5,16,10,22,8,25,10,11,19,3
,12,4,8,4,9,3,24,12,12,11,19,14,13,9,6,
36,15,16,16,19,16,19,16,17,21,7,5,3,17,
17,32,5,13,20,10,25,5,9,5,10,6,24,25,23
,14,13,11,12,11,23,24,13,……
```

Observations:

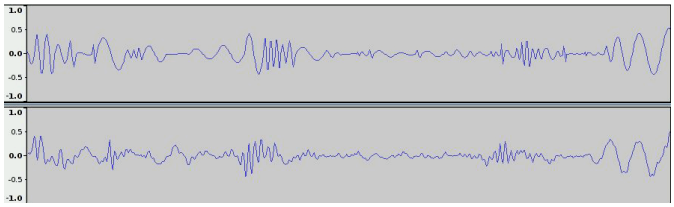
The initial verification of the extraction procedure consisted of comparing audio waveform diagrams with the data file. Particular sections of the waveform were magnified and the corresponding section in the data file was viewed in order to check the accuracy. This process was designed only as a basic test until the synthesis process had been completed, which offered an aural verification method. It was noted that there was a strong correspondence between the data file and the audio waveform, which was expected.

Synthesis

Procedure:

The synthesis process is carried out by a program called *ProcessFrequency.java*. This program uses the data file to create an audio file that synthesises the fundamental frequency. This process works by reading in the sample per cycle value from the data file and generating a sine wave that corresponds to this value. In this way a tone of the same frequency is generated, without the associated timbres and harmonics.

```
while ((sample < freqVal) &&
     (counter < returned)){

  buffer[counter] =(float)(
    (Math.sin(2*Math.PI*((double)sample/(
    double)freqVal)))*maxAmp);
  counter++;
  sample++;
}
```



This graphical representation compares the synthesised wave (top) with the original file. You will notice that some detail is gone, due to the eradication of the timbre.

Observations:

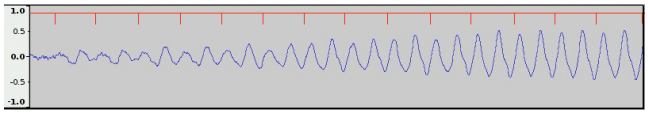
The synthesis procedure produced some very interesting results. Whilst rendering some pop tunes with the process, some new bass notes were heard that were not from the original sound recording. After listening to the original again it was noted that these were in fact the fundamental frequencies of the drums. Listening to the rendered file, it made these frequencies stand out in the original recording. It helped attune my ears to something I had been missing. So now when I listen to the original song again, I now hear the lowest tone of the drum more clearly. The tones are the same frequency, but the rendered tone is much easier to discern with the strong timbre and overtones removed. This acknowledgement also verified the correctness of the extraction and the synthesis procedure.

A problem occurred with the program due to its nature of discarding all data not associated with fundamental frequency. Everything was at the same amplitude. This was not an immediate concern because the amplitude attribute study could be used to correct this. It did however slow things down somewhat. The problem became most evident when a sound was fading out, because what would actually occur was that the synthesised sound would cross fade into noise. In some instances this was quite acceptable, and nothing was done about it, but for some others the amplitude needed to be addressed. Rather than run the amplitude program separately, the frequency extraction and frequency synthesis programs were converted to also incorporate amplitude on demand. That way a new data file could be created which included both amplitude and fundamental frequency data, which would then synthesise a new wave also using both sets of data. This was implemented as a time saver, because the amplitude envelope could have been created and applied separately using the amplitude study programs.

Scaling

Procedure:

There are a couple of programs to carry out scaling techniques. The *ProcessFrequency.java* program explored before in the synthesis process is also capable of scaling. It can scale both frequency and time. The frequency range can be scaled between two predefined frequencies. This method is used for pitch shifting the sample, or indeed stretching it over a much larger range. If you have a recording that covers only an octave, the program can spread the synthesised reproduction over 8 octaves, or indeed any range in the frequency scale. *ProcessFrequency.java* also does time compression and expansion. This is achieved by increasing the number of cycles the program creates, so instead of doing a 1:1 mapping of data to cycle, it will do a 1:n mapping, where n is any real number. This meant that I could stretch any sound infinitely without loss of quality due to stretching. An interpolation/quantisation factor was also added so that the frequency could slide from one point to the next.



2x stretched audio file extract

Using the algorithm above, another time expansion/compression program was created. The *Stretch.java* used the data file created by the extraction program, but it was applied to the actual audio file, rather than synthesised. This means that entire sound may be stretched with its full harmonics and timbre intact. The program goes to two adjacent zero crossing points and cuts that section out as a grain. However unlike a granular stretching technique, the grain is not enveloped. Because the start and end points are already at zero, the enveloping technique is not needed. Also unlike other granular stretching techniques, the grain size is much smaller, consisting only of a single cycle. So this single cycle is captured, and it is repeated as necessary to perform the stretching. If the stretching ratio is 1:430, then the cycle will be repeated 430 times before moving onto the next cycle. This procedure for stretching allows the audio to be stretched an infinite amount without the loss of quality that is common with many stretching algorithms beyond a certain point.

Observations:

The scaling procedure produces some exciting results. The frequency scaling algorithm is useful for pulling apart different pitches when they are very close together, which allows for a closer analysis of the number of different frequencies that are present. In many soundscape recordings, especially those containing running water, or rustling leaves, there is a multitude of similar sounds that can be sifted through. This process makes that easier, and adds a much more musical element to the sound. Instead of 1/16 or even 1/50 semitone separations, the frequencies can be separated by entire tones, or even octaves, turning a rustle into a musical phrase that could be repeated by an oboe.

The time scaling algorithms significantly help to define individual pitches. Rather than a rush of 5000 different fundamental frequencies in one second, the time can be stretched to such a degree that each frequency could last a few seconds, or even minutes so they can be truly recorded and appreciated. Barry Truax pointed out that time stretching is a way of magnifying a sound so that it can be better appreciated and understood. His own time stretching algorithms were used to display the beauty of natural sounds such as waves crashing in slow motion.

One problem that was noted with the procedure however was a quantisation of the frequencies. In many

respects it was not a problem, as the quantisation gave a clear image as to what was happening, but sometimes it also made the audio seem rather disjointed. A quantisation / interpolation factor was included to allow the frequency to slide from one point to the next, over the duration of the scaling factor.

After the success of the time stretched synthesis procedure, it was a natural evolution to try and do this with the direct audio. This process was simply a substitution of synthesised audio with the original audio image. No other measures were taken. As can be seen in the diagram shown previously, the audio file is continuous. The process works very well. There was one problem encountered however, that was not picked up with the synthesised version. A problem I encountered when working on the stretching program was that the pitch would sometimes jump or halve in frequency. The reason for this was that occasionally the timbre of the cycle was very rich and the wave would cross the zero line even though it was not the end of the cycle. A number of measures were put into place to ensure the correct fundamental frequency value could be attained. The first step was to set up a more variable window length, so instead of capturing just a single cycle, the composer could tell it to capture 3 cycles in one block. That way there was a manual override if the program was acting irrationally. The second procedure was to automate the process. The program would look ahead to see if there was an erratic jump and automatically decide whether it needed to capture another cycle or not. This process has proven to work well, as long as it is kept within a small set of limits. If the program is left to wander on it own accord it sometimes starts counting over 30 cycles per value, before it starts making its way down again. This creates an echo event, which is not desirable. It was capped at 13 cycles maximum, and 1 minimum. A hard limit of just 2 to 3 cycles also works well.

Substitution

Procedure:

For the substitution procedure, I made a variant on the *create amplitude envelope* function from the amplitude study. I added a procedure to it that would find the highest and lowest points in the file and used these as the limits for an amplitude envelope, which would then scale the frequency data to fit an amplitude envelope.

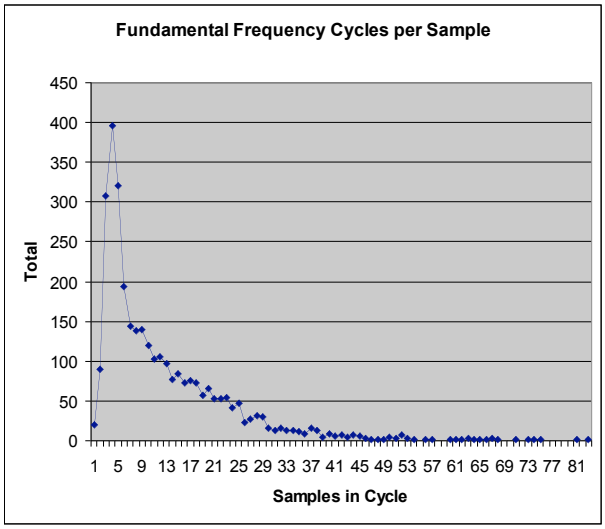
Observations:

The computer process of creating the amplitude envelope is twice as long when using frequency data, due to the double pass nature of the program, but the process is still fast enough that it seems a negligible comment. The amplitude data created was quite extreme depending on what frequency data was used. Overall the process worked, but finding a use for the frequency as an amplitude envelope was a daunting task. Some parts worked well, others not. It was more a matter of trial and error at this stage to see if anything usable would come of it.

Generation

Procedure:

The generation process uses a procedure of post-analysis, and would be considered a second level of abstraction. The purpose of the generation procedure is to find the fundamental elements of the structure, so that other structures can be created, which are not identical, but which share the same characteristics, and would be recognized directly as an extension of the original recording. This process was achieved by running a data file through a spreadsheet program and measuring the mean and standard deviation of the data, enabling the data to be described in a mathematical language. Using this information a model of the audio event is created which generated frequencies using a Gaussian distribution with the mean at the centre, and the standard deviation as the bandwidth. This method creates a frequency data file, which can then be used with any of the procedures mentioned in this paper that use a data file.



This graph of distribution peaks at 5 samples per cycle. The Mean is 13.1849, the standard deviation is 10.67322. There is still a large range of figures outside these values, which can be accounted for using an abnormal deviation spread. In the case of the diagram given, the most effective model would be made by setting the mean at 5, and using a spread from 1 to 84 as the standard deviation, using a curve similar to the graph, rather than a Gaussian distribution. The rise of the curve is a very steep straight line, whereas the fall-off is definitely an exponential function. This method requires a new model for every single audio sample.

Another tool for generation was to create a data mining tool which scanned the file looking for significant pieces of information. It looks primarily for repeated patterns, and mostly repeated patterns. The patterns returned to a data file, and can be used as small motifs at the composer's discretion. A brief sample of the output is given below.

```
Exact Matches: 2, containing 11 sequential
samples. The sequence was:
4 3 3 2 3 2 3 3 3 3 3
Exact Matches: 2, containing 11 sequential
samples the sequence was:
3 3 2 3 2 3 3 3 3 3 7
-----------------------------------
The most popular 12 number sequence/s oc-
curing 2 time/s:
4 3 3 2 3 2 3 3 3 3 3
3 3 2 3 2 3 3 3 3 3 7
-----------------------------------
Exact Matches: 2, containing 12 sequential
samples. The sequence was:
4 3 3 2 3 2 3 3 3 3 3 7
-----------------------------------
The most popular 12 number sequence/s oc-
curing 1 time/s:
4  3  3  2  3  2  3  3  3  3  3  7
-----------------------------------
All matches found. Sequence is ending
```

Observations:

The generation procedure of using a Gaussian distribution works well for patterns that already seem quite random in nature, such as rivers, and rustling, and even whistles. If however one uses, for example a pop tune as the input, the outcome is far removed from the input, there are far too many different sounds involved, not to mention durations and relationships between notes and frequencies, which this kind of analysis does not take into account. As this project is focused on the sounds of nature this is a suitable outcome. If the project were based on creating generic pop tunes using this kind of analysis, it would be a dismal failure.

The data mining experiment for extracting motifs has found some interesting groups of numbers, although further investigation into their use within the larger scale is still something that is in progress. So far these significant numbers have been used to create musical structure, such as numbers of bars, and melodic phrases. They have not been used in any automated function for the computer. They have been left over for the composer to work with in a more creative fashion. It is intended that they will play a more crucial role in the final outcome of the attribute study, most likely as an overarching structural device, but so far there have been no overarching structural devices put into play, as there is still space and timbre to work with before all parts of the study will come together.

Conclusion

This paper has outlined the aims of eco-structuralism, and has expanded on the idea of fundamental frequency as a structural device. Fundamental frequency is part of an attribute study that involves the search for structures within audio files. A number of different procedures have been outlined that display how a fundamental frequency structure can be extracted, and what can be done with it. All of the structural devices and procedures mentioned fit into the realm of eco-structuralism in a formalised and cohesive manner. They give the composer more control of the audio structure with which they may compose.

References

Dahlhaus C. 1982. *Realism in Nineteenth-century Music*. Cambridge University Press.

Field, A. 2000. "Simulation and Reality: The New Sonic Objects" *Music, Electronic Media and Culture*. eds, Simon Emmerson. Aldershot: Ashgate Publishing Limited.

Gibson, J.J. 1966. *The Senses Considered as Perceptual Systems*. Boston, MA: Houghton Mifflin

Hodgkinson, T. 2001. "An interview with Pierre Schaeffer" *The Book of Music & Nature*. Wesleyan University Press.

Keller, D. 1999. *touch'n'go: Ecological Models in Composition*. Master's thesis, Simon Fraser University, Burnaby BC.

Keller, D. 2000. "Compositional processes from an ecological perspective" *Leonardo Music Journal*. 10. 55–60.

Keller, D. 2005. *Ecocomposition: Where does it come from? Where is it going?* Program Notes, College of Music, University of North Texas.

Keller, D. and Capasso, A. 2006. "New concepts and techniques" In eco-composition in *Organized Sound*. 11, 1. Cambridge University Press.

Keller, D. and Truax, B. 1998. "Ecologically-based granular synthesis" *ICMC Proceedings, 1998*.

Kivy, P. 2002. *Introduction to a Philosophy of Music*. Clarendon Press.

Opie, T. 2005. "Amplitude Analysis in Musical Composition" *ACMC 2005 Generate & Test Proceedings*. Brisbane.

Opie, T. 2006. "An Introduction to Eco-Structuralism". To be published in *ICMC 2006 Proceedings*, New Orleans.

Schafer, R.M. 1994. *The Soundscape: Our Sonic Environment and the Tuning of the World*. Destiny Books.

Schoenberg, A. 1975. *Style and Idea*. Belmont Music Publishers.

Truax, B. 1994a. "Discovering inner complexity: Time shifting and transposition with a real-time granulation technique" *Computer Music Journal*. 18, 2. 38–48.

Truax, B. 2001. *Acoustic Communication*. 2nd ed. Westport CT: Ablex Publishing.

Truax, B. 1994b. "The Inner and Outer Complexity of Music" *Perspectives of New Music*. 32,1. 176-193.