RAAPT: Rhythmic Automated Audio Processing Templater

Researcher: Ian Rios, Faculty Sponsor: Professor Robert Hamilton, HASS Department

Sampling

In music, sampling is the act of taking a portion, or sample, of one sound recording and reusing it as an instrument or a sound recording in a different song or piece. In the figure below (fig. 1), you can see a graph of amplitude over time for a recording of outdoor noises. When graphed in terms of amplitude over frequency, the graph shows peaks in the low bass range (10-100 Hz), shown in red and light blue/green in the spectral graph below (fig. 2). These areas interest us most because they can be matched to bass drums and low lead instruments. As you can see, the peaks in the overall amplitude do not directly correlate the peaks in the 10-100 Hz range. The peaks not represented in figure reside in different audio ranges, which can be heard by using a different FFT window, described below.

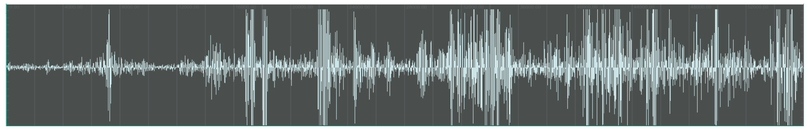

Figure 1: Amplitude over time

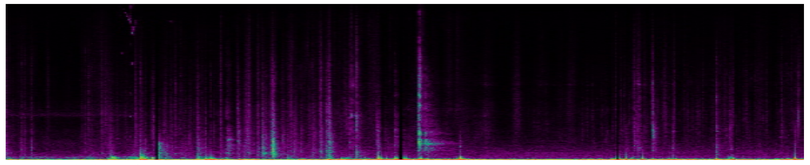

Figure 2: Amplitude times frequency over time

Desired Outcome

For this project, I recorded multiple soundscapes of varying energy within the spectrum of audible auditory experience (fig. 3). Using this recorded audio, the program will decide what samples within the original file sound most like real instruments by pattern matching them to frequency bands of the particular instrument. Using a fast Fourier transform algorithm to output amplitude values for specific frequency ranges (fig. 4), we can accurately list the ideal time values to analyze when resampling later to create an instrument.

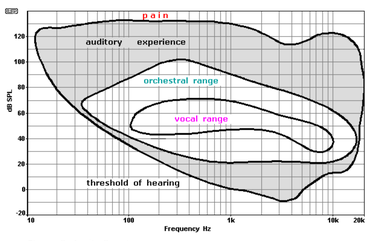

Figure 3: Audio Spectrum

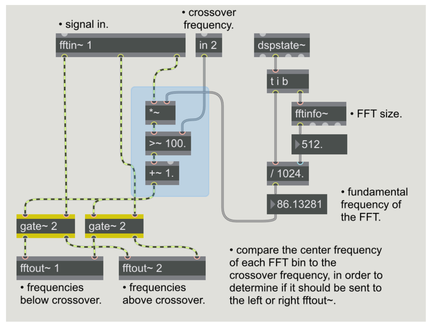

Figure 4: FFT Algorithm

To decide what values would be ideal, the program takes a running average of amplitude over four frequency bands, with a window size of 1000 milliseconds. For those values, the user has the option to select a range from high amplitude and very few samples, to low amplitude and all available samples over the entire file size, to allow for dynamic gain control. In the sub patch on the right (fig 5.), the user inputs a float that decides the range that the maximum amplitude should be. These amplitude values are saved to be reprocessed later when generating start points for instruments.

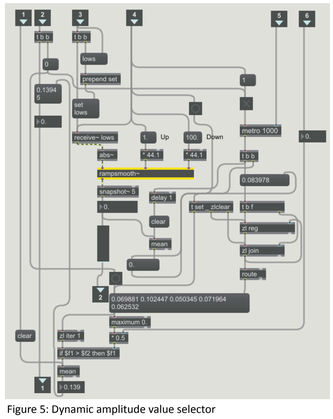

Figure 5: Dynamic amplitude value selector

Abstract

The RAAPT project generates rhythmic musical structures by assigning pre-made templates to ecological soundscapes. By applying structural templates to audio clips, RAAPT creates an automated and generative music system capable of transforming any sonic material into rhythmic, musical instruments.

By choosing start and end points within an audio file and selecting frequency ranges to pre-existing instrument frequencies, the program identifies which sounds are the best matches for predefined patterns.

Drum samples and lead sounds are extracted from the source audio by matching regions to a specified amplitude and frequency range. By shaping the sound to match different waveforms, such as a kick drum or hi-hat, a bank of drum samples is automatically generated. A granular synthesizer processes the same audio file and creates a bank of lead samples that are used as melodic elements within a piece of music.

All samples are then run through effects and stored in a post-processed sample bank.

Finally, a sequencer organizes all the drum and lead sounds and maps them to MIDI presets chosen from a bank of MIDI loops.

The RAAPT project can generate multiple versions of the same piece of music from any raw audio file due to non-deterministic properties of selecting sample length, location, and effects. This approach decouples the composition of musical structure from the traditional orchestration of voices and instruments. Composers using RAAPT can approach creating new material in exciting and experimental ways.

Data Storage

This program takes O(2n) time to run, where n is the length of time of the input file. If you were to process an audio file of length 10 minutes, it then turns into a 20 minute operation, which is not realistic for live performance and manipulation of sounds. To overcome this, I created a database structure (fig. 6) to store project files of preprocessed audio. A database entry stores filename, size, predefined user analysis points and post processed chosen frequency values. By loading a project file instead of a sound file, the user can bypass the audio spectrum analysis section and go straight to the generation of samples to create music.

```
newProject*
File  Edit  View  Object  Arrange  Options  Debug  Extras  Window  Help
1, elevatorDown2.wav;
2, 56000.;
3, 0.207051;
4, 0.019851;
5, 0.013853;
6, 0.004056;
7, 48776 2982 4445 5999 6025 11136 12872 13874 14621 19233 53014;
8, 5974 12872 13825 14587 19220 24560 25350 36535 48872 52999 53019;
9, 2464 5973 12872 14593 20280 24344 25365 35996 36511 48871 52999 53019;
10, 31364 41368 48662 51822 52995 53001;
11, 1783 2774 5714 7573 15075 17750 18679 21303 23573 31519 32470;
12, 5975 13826 14572 19235 24577 25303 36513 48866 52995 53020 54575;
13, 1779 2752 5701 7575 15077 17753 18675 21283 23562 30916 31526;
15, 0.825;
16, 0.825;
17, 0.875;
18, 0.55;
```

Filename
Duration
Output amplitude per frequency band
Generated ideal start times
Ideal amplitude range per frequency band

(Figure 6: Database entry

After the audio file gets processed, the start location, end location and total duration values are chosen for each individual instrument (fig. 7). These are sampled from the original file by using the amplitude values generated in the previous step. Since these values are chosen at random from a list, the user can choose another one in the event they don't like the one the program chose.

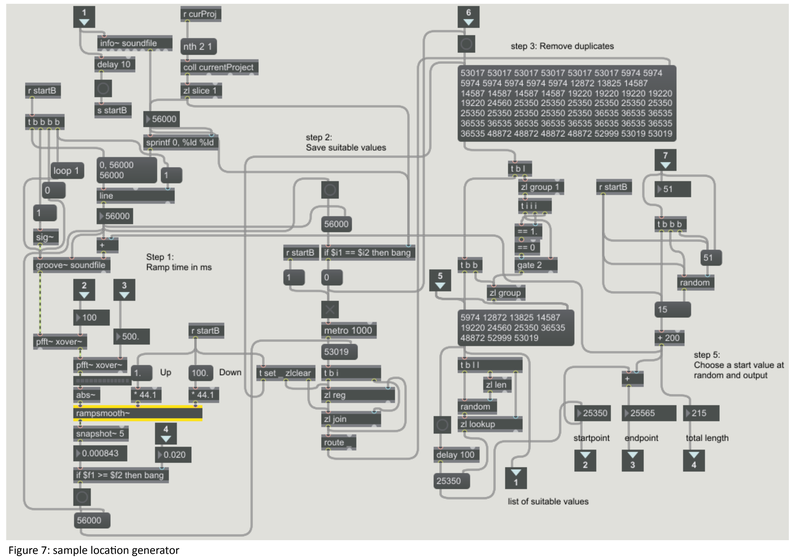

Figure 7: sample location generator

Instrument Shaping

To create realistic sounding instruments using the newly defined start and end times, I cross reference the frequency analysis of known instruments with the waveform of the sample that we generated to ensure that the amplitude over frequency matches the desired instrument. In figure 8 we see the frequency spectrum of a snare drum with a resonant frequency of 450 Hz at –6db and a noise threshold from 1 kHz to 20kHz below –12db. Below it in figure 9, we see the frequency analysis of the program generated sample, which is resonating at 250 Hz at –15db and has a noise threshold from 700 Hz to 17 kHz at –25db. This generated sample fits the criteria for a snare drum, and is in turn processed in the right sub patch (fig. 10) to ensure a loud and clear enough output volume while retaining a similar shape to the natural snare drum using dynamic EQing.

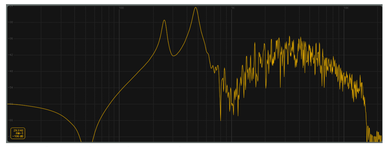

Figure 8: Natural snare drum sample

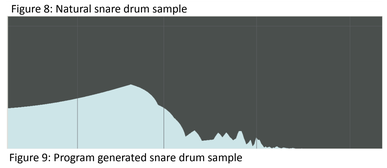

Figure 9: Program generated snare drum sample

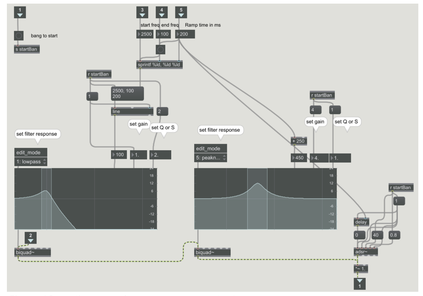

Figure 10: audio correction to ensure sample shape

Recording

In order to guarantee every sample be heard (as some samples may fit the frequency criteria but have small amplitude values), compression is used to raise all the samples to an ideal threshold for playback (fig. 11).

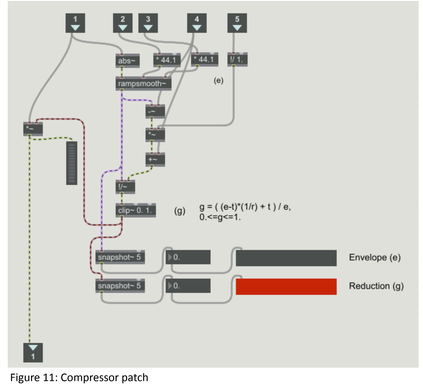

Figure 11: Compressor patch

The post compressed audio is then recorded by looping the segment of desired audio from the main file once and storing the sample to a unique buffer (fig. 12) to be used in the final step. This operation takes O(k) time where k is the duration of milliseconds of the desired sample.

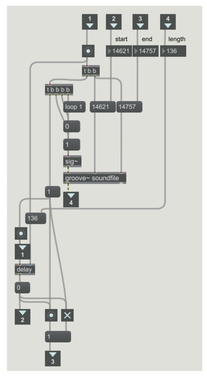

Figure 12: Recorder patch

Granular Synthesis and Playback

Granular synthesis is a basic sound synthesis method that operates on the microsound time scale based on the same principle as sampling. The samples are split into small pieces of around 1 to 50 ms called grains. For this project, the user has three lead grains that have been selected based on frequency activity in three FFT bands. The grain samples may be layered on top of each other, and may play at different speeds, phases, volume, and frequency, among other parameters (fig. 13).

For playback of the drum sounds, I created a 16 step rhythmic drum sequencer that has options for random probability or MIDI loop or file playback for music production. The user can also program live sequences easily for performances.

The end result of the program proved to be successful in taking any audio file and outputting instrument samples that could be manipulated in real time to generate music.

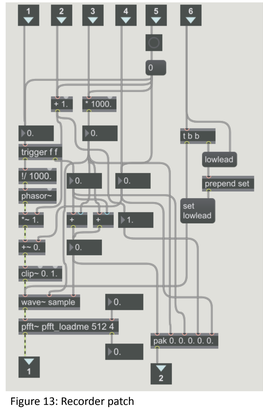

Figure 13: Recorder patch