Parameter automation for granular synthesis

author: Karol Bakunowski

supervisor: Michael Zbyszyński

BSc (Hons) Music Computing
Goldsmiths, University of London

2019

Abstract

My research is about stuff.

It begins with a study of some stuff, and then some other stuff and things.

Acknowledgements

Acknowledge all the things!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Aims and Objectives

The initial aim of the project was to implement a machine learning solution to the task of granular synthesizer programming based on sound matching. Consequently building a tool that would assist musicians in creating interesting sounds, provided an audio input to the system.

Over the course of the past couple months, the project extended into a more technical-oriented direction, implementing a granular synthesis algorithm in the 'JUCE' C++ framework. As well as into a research-oriented space, with a focus of finding the best possible audio descriptors for specific characteristics of sounds. In order to determine the most significant in judging similarity on the cognitive level.

The methods here concerned with describing audio differ from other approaches in literature (SOURCES)primarily in that sound is treated in a modular manner, where a description is made based on one characteristic for X number instances, and separate processes are run for each. Instead of trying to describe audio as a whole, the aim is to describe it as a combination of things (density, pitch, rhythm, amplitude).

Concretely my objectives are:

1. To build a tool that is helpful to artists in creating new sounds

2. To challenge the interaction between an artist and a preset as a starting point to synthesis

3. To achieve a response that is not only stimulating to the user but also differs from simply randomizing the parameter values

In measuring the project's success, the subjective sonic coherence and similarity of algorithm's outputs may be considered the best indicator (the human discriminator?), along with more quantitative analysis, such as comparing the audio descriptors on input and output sounds for example.

In (future chapters) I provide some critique on established techniques of assessment of these types of problems and ultimately conclude that some stuff in the best way of doing it.

1.1.1 Deliverables

To these ends I have implemented 3 separate pieces of software:

- Granular synthesizer implemented in the "JUCE" framework in C++
- A module responsible for analysing audio input, and sending that data
- A mixture of machine learing algorithms implemented in Python, that actually change synthesizer's parameters based on the input sound

Chapter 2

Literature Review

2.1 Problem background

Programming synthesizers is a fun, but challenging task. The range of sounds possible to achieve on such a device, and consequently the amount of adjustable parameters can be overwhelming. From choosing an audio file to sample, to an amplitude envelope for each grain, the ability to make decisions about programming a granulator has to come from either a place of certainty about what each parameter is responsible for, or a place of experimental thought, and a somewhat random parameter value assignments.

Users fairly new in the realm of synthesizer programming may encounter issues creating sounds they desire. Even successful musicians might be doing things following a not clearly defined "intuition". There are of course people who are experts in this field, but artists often look for inspiration when it comes to timbre of their sounds. It would seem that, the use of presets as a starting point is not unusual.

Research has been done previously as an investigation into automation of parameters in synthesizers based on sound matching. Taking a snippet of sound, the algorithm would try to find parameter settings to match a produced sound as closely as possible to the source

This research mostly focuses on FM synthesis , although experiments on different synthesis techniques has been done , including any VST plug-in

Also, it seems that most of this research focuses on reproducing the original input . Perhaps more interesting and novel sounds could arise as an effect of bad performance, but is does not seem to be the desired outcome in most cases.

However, using this approach on corpus-based synthesis remains an untapped area of research, worth investigating

2.2 Audio descriptors

MFCCs are a very popular descriptor used widely in the domain of speech synthesis. Some literature also uses it as the means of comparing two sounds together(cite matthew, or some other study). It is probably the most sophisticated algorithm for describing timbre available currently.

There is of course a huge variety of audio descriptors available, and some serve different purpose than others. One can get very specific and use descriptors for onsets, pitch, etc.

These really gain power when used in parallel, or together, rather than on their own. That is the main reason behind the approach of describing audio in this project.

Especially when trying to describe audio to a machine - an agent of you will, that will be responsible for the programming of the granulator. Thinking about it in terms of how humans would approach it, one would need to describe specific characteristics of a particular sound to replicate it. Often this happens 'behind the scenes', as our perception does most of the work for us, and we never really explicitly think about all the separate trains of information that we have to process in order to achieve an understanding of sound.

In this specific case, we are approaching the programming of a granular synthesizer, therefore we need specific metrics of describing sound that can translate well to the parameters of a granular synthesizer.

2.3 Granulation

Automating parameters for granular synthesizer can be a difficult problem, considering how heavily the output sound depends on the audio being sampled. In other words the source of the grains.

To accommodate for that difficulty, some universal audio descriptors have to be used in order to describe the sound to the previously mentioned agent. These include (change when done) descriptions of rhythm, density, pitch etc, so that the input sound can be 'molded' into what will resemble the original sound that the agent is trying to replicate.

Concatenative synthesis - another thing widely used in speech synthesis offers a solution to this problem by creating a 'pool' of grains to choose from in order to replicate a sound (offer better description). It could be an interesting way to approach this problem, however a lot of research has been done in this domain (CataRT) and it seemed less experimental to tinker with. (the truth, but obviously change later)

2.4 Machine Learning

A sophisticated neural network could be implemented to essentialy build a granular synthesizer parameter space form the ground up using only the input sound, however implementing such a system and predicting it's behaviour is a difficult task as well as a very experimental approach, that is quite difficult to control and predict.

Focusing on replicating certain aspects of the input sound seems like a more straightforward and most importantly predictable approach to this problem.

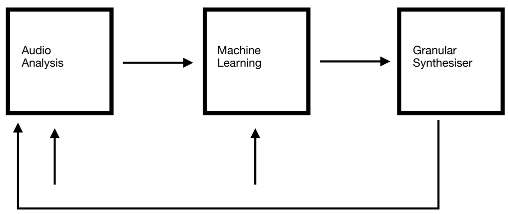
Investigating the correlation between certain audio descriptors and their influence over specific parameters of the synthesizer, and consequently separating the machine learning tasks to predict those aspects of the sound separately will result in a much more predictable outcome, as well as one that's easier to control.

Chapter 3

Methods

The method of tackling this problem will be based on previous research done in this domain. There are three essential components for this project:

Figure 3.1: Basic flowchart



3.1 Synthesis

- how it is build

- what libraries etc are used

- what parameters are being manipulated

- how each parameter influences the output sound

- what sounds are being sampled (where do the grains come from)(how does the behaviour change depending on the sample)

The box “Granular Synthesiser” in figure 1. represents a granular synthesizer. It is build in C++, using the 'JUCE' framework(SOURCE). (Add about it being quasi-synchronous, and all other possible information.)

3.2 Audio descriptors

- how it is build

- what libraries etc are used

- what testing has been done

- what descriptors influence what parameters

- how do they work (maybe briefly)

- how do they speak to the 'agent'

The “Audio Analysis” box in figure 1. represents the aspect of the project responsible for extracting audio features from input sounds. This requires some testing in order to see which works best for this task. Feature extractors considered are: MFCC and DBM. The environment of choice is the “Essentia” library for C++. (SOURCE)

3.3 Machine learning

- how it is build

- what libraries etc are used

- how is it trained

- what data is used

“Machine Learning” in figure 1, stands for Machine Learning. This aspect will most likely be implemented in python, either using the “scikit.learn” library, or the “TensorFlow” library. Depending on which Machine Learning algorithms will be used. This will be determined during testing. The ones being considered at the moment include k-NN, Neural Networks, and LSTM(Long Short-Term Memory Neural Network) Training of whichever algorithm chosen will be happening on data generated by me. Vectors of synthesis engine parameter values, and audio analysis values, possibly as a CSV file will be created. The goal here is to automate a sampled walk through the parameter space, and extract audio features for each point. Possibly, a genetic algorithm could be used for this process

Chapter 4

Results and Analysis

- statistics, tables and graphs

- qualitative as well as quantitative data

- comparison between random button, and the algorithm

- user testing

- questionnaires for users (friends, or reddit friends)

- no interpretation of results

Chapter 5

Discussion

5.1 Summary of Findings

5.2 Evaluation

5.3 Future Work

Chapter 6

Conclusion

.1

Title of Appendix A