AALBORG UNIVERSITY

A Generative Voice Driven Percussion Application

Master Thesis

By

Iakovos Vogiatzoglou

Department of Media Technology

for the degree of
Master in Sound and Music Computing

Master Thesis Supervisor:
Hendrik Purwins
Department of Architecture, Design and Media Technology
Aalborg University, Copenhagen

Date
27/05/2016

ABSTRACT

The research and implementation of an interactive music application prototype that maps human voice sequences, such as beatbox rhythms into drum samples, is documented in this master thesis project. In addition to the mapping and transformation of the human voice input, a continuation system that predicts and generates new patterns according to the same style as the input is implemented.

TABLE OF CONTENTS

Chapter Page

**********2

CHAPTER 1: INTRODUCTION

The human voice can be considered as an immensely expressive musical instrument. It allows even a non-musically trained person to easily produce musical sounds by exploiting the features of the vocal tract. This project attempts to utilize the qualities of the human voice by creating an application that maps percussive patterns (beabox) to actual drum sounds. In addition to the mapping of the beatbox sequences to synthetic drums, a sequence prediction and generation technique was implemented, inspired by (Pachet, 2003). The system analyses the incoming beatbox signal and attempts to generate a new sequences in the same style as the input. Although the current implementation is focused mainly on percussive patterns, future plans are to enable the application to work with humming and whistling, making it a complete sonic sketchpad for music production.

TimbreID, a timbre classification library for Pure Data (PD), a real-time graphical programming language was used for extracting features of the beatbox rhythms and training of synthesized drum samples. Additionally a Variable Length Markov Model (VMM) package for Python was used for predicting and generating new patterns in the same style of the input patterns of the user.

**********2

CHAPTER 2: BACKGROUND

In this section an investigation on the role of the human mouth in music applications and Music Information Retrieval (MIR) systems is presented, along with all the necessary background information for the employment of this project, as well as past work and state of the art applications in the relevant fields. Topics include some of the most popular methods for Timbre Analysis and Sound Classification, Onset Detection, Sequence Prediction and continuation as well as a brief description of the Open Sound Control(OSC) protocol, which is vital for communicating between different software platforms.

2.1 The Human Voice as a Musical Instrument

Using the human voice in a musical way is not only limited to singing or whistling. Beatboxing is a prime example, where the mouth is used in order to imitate musical sounds such as percussion, but not only limited to. Although there is not a lot of academic research on the topic, (Stowell, 2010) attempts to document in depth some of the most important beatbox characteristics but also specific techniques that are used from beatbox artists.

Beatboxing is prominently connected to hip-hop culture and its roots date back to the early 1980s, where drum machines made their first appearance in the music production world.

2.2 Past Work

In this section various projects that use the human mouth for the creation of sound generating or machine learning applications will be presented. Janer (Janer, 2005) created a voice driven synthesizer in Pure Data, where the human voice controls Frequency Modulation (FM) synthesizer parameters. Hazan (Hazan, 2005) developed a real time voice driven drum generator using the BillaBoop VST Core plug-in Sinyor et al (Sinyor, 2005) developed a beatbox classifier (kick drum, open hat, closed hat and two types of snare drum) using the Autonomous Classification Engine (ACE). Stockwell et al (Stowell D. a., 2010) developed a beatbox dataset in addition to their beatbox classifier.

2.3 Feature Extraction Techniques

On this section some of the standard techniques for extracting features from audio speech signals will be described. Some of these techniques are generally used in speech recognition systems but also in percussive timbre

identification applications. While looking at a waveform in the time domain can provide useful information about time and amplitude, when it comes to extracting information for classification we want to be able to decrease the number of features and extract the most important ones (Shrawankar, 2013). A summary and brief description will be given for some of these feature extraction methods, although emphasis will be given on the Mel Frequency Cepstrum Coefficients (MFCC), as this is the method that was practically tested and used on the implementation of this project. Furthermore, several studies found MFCC to give superior results to the Linear Predictive Coding (LPC) and Linear Predictive Cepstral Coefficients (Gulzar, 2014), (Dave, 2013), (Mehta, 2013).

Listed below are some of the most popular feature extraction techniques that are used in speech recognition and music modelling (Shrawankar, 2013):

- **Linear Predictive Analysis (LPC)**
- **Linear Predictive Cepstral Coefficients**
- **Mel-Frequency Cepstral Coefficents (MFCC)**
- **Perceptual Linear Predictive Coefficients (PLP)**
- **Power Spectral Analysis (FFT)**

- **Relative Spectra Filtering of Log Domain Coefficients (RASTA)**

Linear Predictive Coding (LPC)

Linear Predictive Coding (LPC), is one of the most dominant techniques for analyzing speech, mostly due to its ability to encode speech at low bit rate. LPC is modeled after human speech and is produced by utilizing a filter model that emulates the characteristics of the human tract. By minimizing the sum of the squared differences between an original speech signal and the estimated signal, a number of predictor coefficients can be produced. It is possible to estimate these coefficients by using a frame of approximately 20 milliseconds (Shrawankar, 2013). Some of the parameters used for evaluating performance include Bit Rates, Overall Delay of the System, Computational Complexity and Objective Performance Evaluation (Shrawankar, 2013).

Linear Predictive Cepstral Coefficients (LPCC)

Another popular approach for parameter estimation is the Linear Predictive Cepstral Coefficients technique. The concept behind this technique involves the prediction of one speech sample at a current time, as a linear

combination of past samples (Gulzar, 2014). A first order high pass filter is used on the input signal for pre-emphasizing, since there is more energy concentrated in the low frequencies. Furthermore, a hamming window is preferred, due to its very low side lobe characteristics. The final stage of the LPC algorithm is Cepstral analysis (Gulzar, 2014), a concept which will be thoroughly described in the next paragraph.

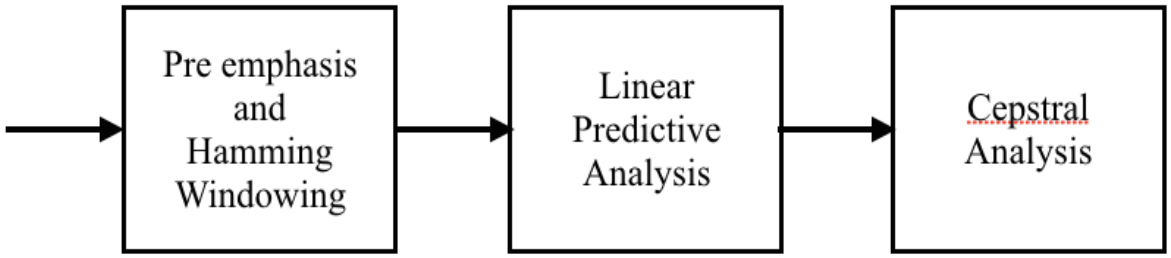

Figure 1 - LPCC Algorithm

Cepstrum and the Mel Scale

Before going through the explanation of the Mel-Frequency Cepstrum Coefficients (MFCC) , a brief description will be given on the concepts of Cepstrum and Mel scale, since they are critical for understanding MFCC and BFCC.

Cepstrum

The term Cepstrum was coined by its inventors Bogert, Healy and Tukey in 1963. The cepstrum can be described as the spectrum of a logarithmic spectrum. Although the original intention for it was to be used for determining seismic signals, it can be used for detecting any periodic structure (Randall, 1981). Initially it was first defined as the power spectrum of the logarithmic, however it was later changed to the inverse Fourrier transform of the log power spectrum, after the introduction of the Fast Fourrier Transform (FFT) (Randall R. B., 2012). An important benefit of this change of definition is the fact that it is easier to change from a function of frequency to a function of time by using the inverse transform (Randall R. B., 2012).

The cepstrum can be defined as:

$$C(\tau) = \Im^{-1}\left[\log\left(X(f)\right)\right]$$

Figure 2 - Cepstrum Definition

Mel Scale

In1937 Stevens, Volkmann and Newman performed a series of experiments for measuring pitch. Their research revealed that humans perceive pitch

linearly below 1 kHz and logarithmic above (Stevens, 1937). The pitch of pure tones can be subjectively defined and for each tone a subjective pitch can be measured on the "Mel Scale". A 1 kHz tone, 40 dB above the human hearing threshold corresponds to 1000 mels (Imai, 1983)

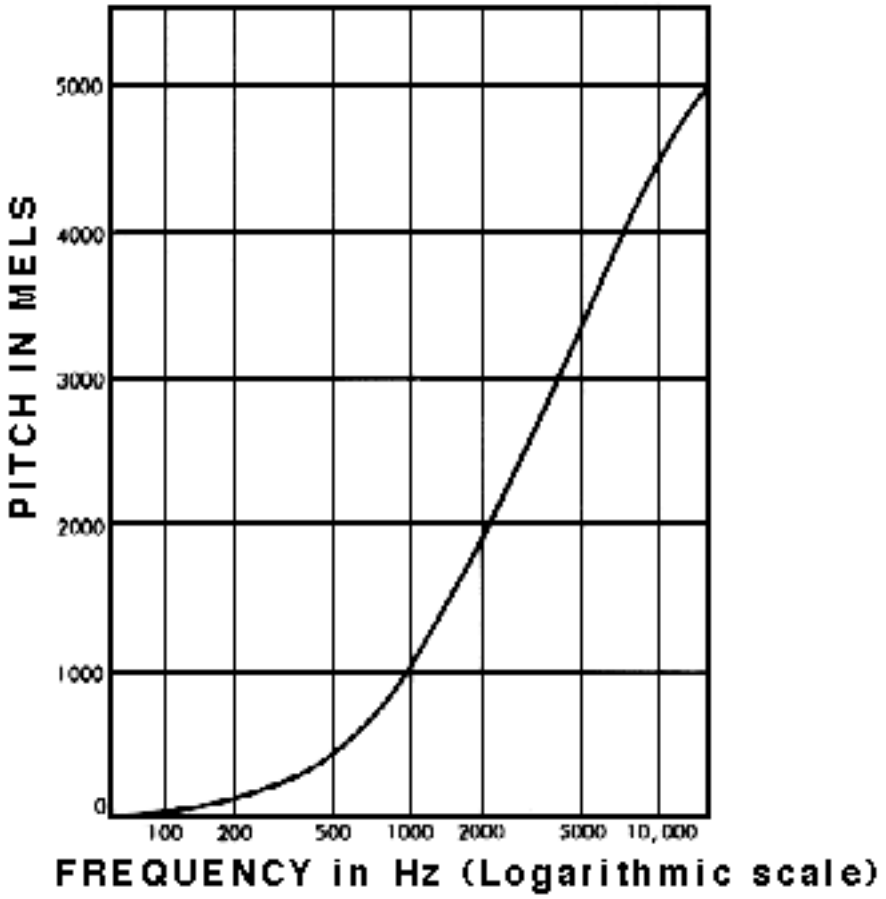

Figure 3 - Mel Scale

Mel-Frequency Cepstrum Coefficients (MFCC)

MFCC are amongst the most popular features for speech recognition and music modeling and processing (Logan, 2000). They are considered to be short-term spectral based features and are based on the human ear "filter" characteristics (Hasan, 2004). The process begins by separating the input signal into smaller frames, using a Hamming window. A spectral feature vector is then generated for each frame (Logan, 2000). The logarithm of the amplitude spectrum is then retained, by taking the Discrete Fourrier Transform (DFT). A series of overlapping triangular bandpass filters are used, spaced according to the mel scale (Brent, 2009). Finally, the Discrete Cosine Transform (DCT) is taken of the log filterbank. We only keep approximately 13 cepstral features for each of the frames (Logan, 2000).

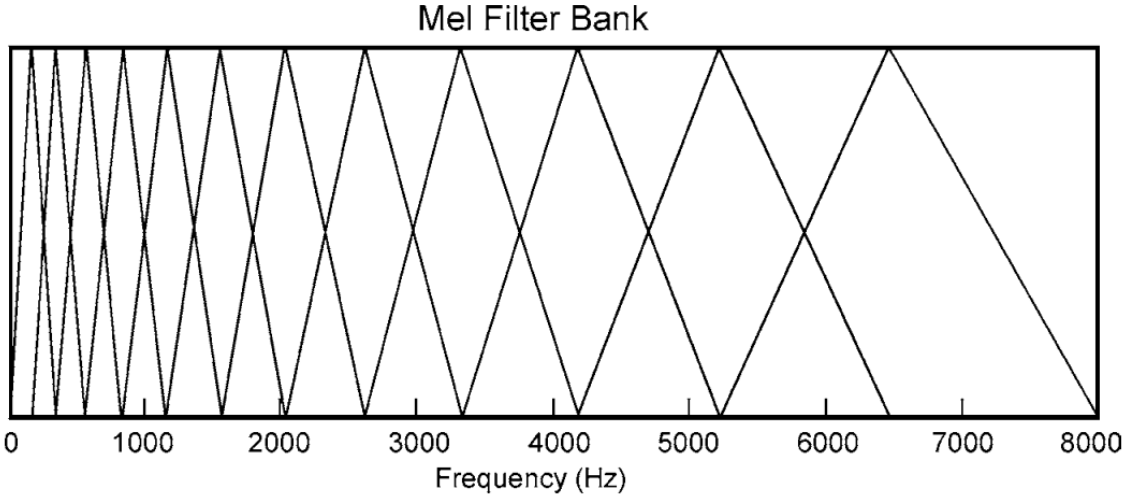

Figure 4 - MFCC Band Pass Filters

**********2

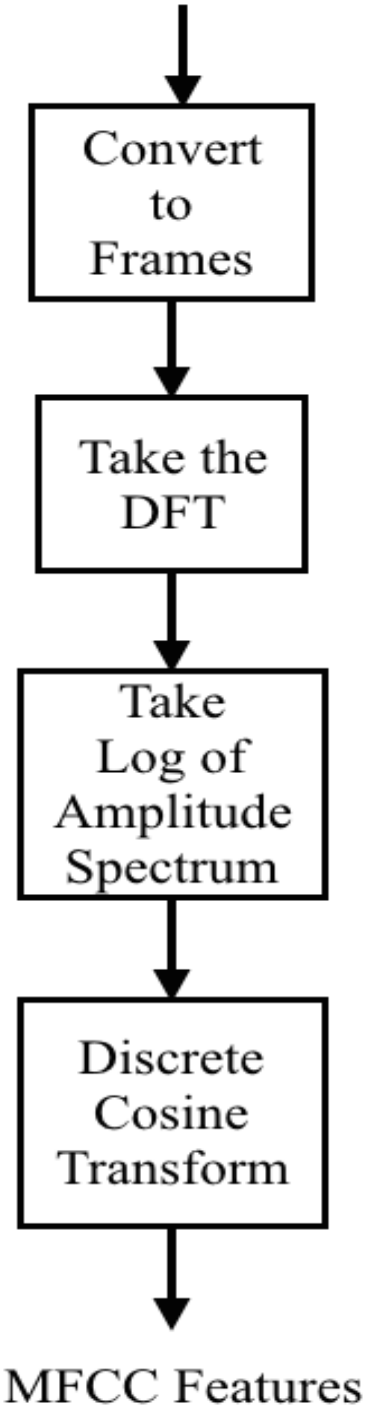

Figure 5 - MFCC Features

P

**********2

erceptual Linear Prediction (PLP)

Perceptual Linear Prediction (PLP) shares similarities with the LPC method as it is based on the short-term spectrum of speech. It was created by Hermansky in 1990 with the intention of describing psychophysics of the human hearing and its main difference fom LPC is the modification of the short-term spectrum of speech (Shrawankar, 2013) .

Power Spectral Analysis

Power Spectral Analysis (FFT) is one of the most popular methods for the analysis of speech signals. In this technique the frequency content of a speech signal is analyzed by the power spectrum. The Discrete Fourier Transform (DFT) is first computed for the obtaining the frequency information of the signal (Shrawankar, 2013).

Relative Spectra Filtering (RASTA)

Relative Spectra Filtering (RASTA) works by using band pass filtering on the feature coefficients in the log domain. It can be used as a compensation for linear channel distortions that appears in the log spectrum. The high pass section of the band pass filter can minimize the possible noise that appears in the convolution process (Shrawankar, 2013).

2.4 Onset Detection

In this section the concept of Onset Detection will be explained as well as critical functions an algorithms of the process. Onset detection of musical signals is critical step when analyzing music and want to examine rhythm and tempo structure. In essence we are looking for the start point of an event, the "transient" of an audio signal. Bello et al, (Bello, 2005) describe the transient as "A sudden burst of energy, a change in the short-time spectrum of the signal or in the statistical properties." Before going through further details on the subject it is important to define three key concepts, the transients, onsets and attacks.

Three Principal Onset Detection definitions (Bello, 2005)**:**

Attack:

The attack of a musical note is the fragment of a musical note where the amplitude of the envelope starts to increase.

Transient:

The transient of a musical note is the part between the excitation and the fading out of the note and is formally defined as: “Short intervals during which the signal evolves quickly in some nontrivial or relatively unpredictable way (Bello, 2005).

Onset:

The starting point of the onset is right after the transient. There is an overlap between the two.

Onset Detection algorithms can be categorized into a *detection function* or a *peak picking stage* (Duxbury, 2003). Detection function makes the discovery of onset transients easier by converting a signal to a function. The Peak picking stage refers to the localization of the detection function that is corresponding the onset transients (Duxbury, 2003).

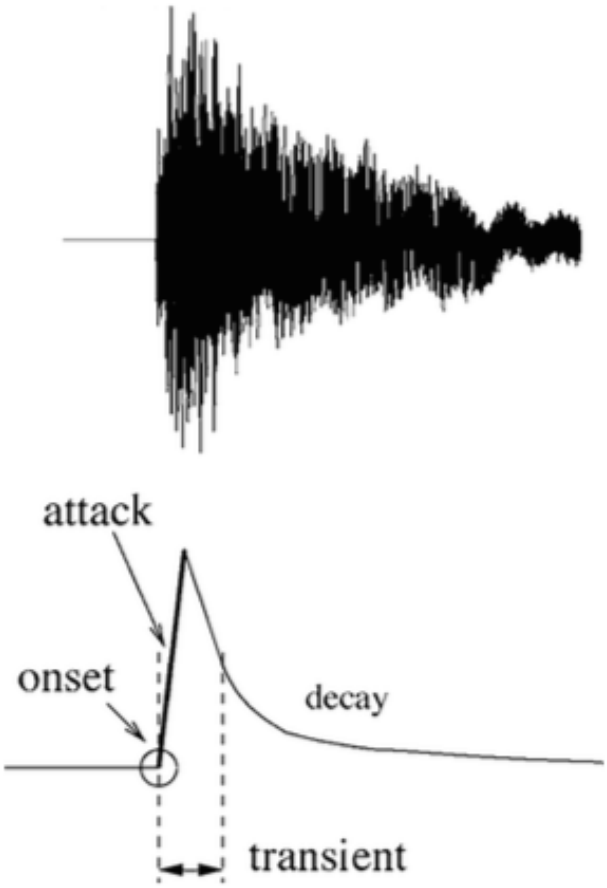

Figure 6 - Attack, Transient and Onset, Bello et al

2.5 Sequence Prediction and Generation

The use of machine learning algorithms for music composition is a rather popular practice (Dubnov, 2003). Machine learning algorithms can be used for modeling stylistic features of certain music compositions but also for predicting new music patterns from a given input score. It is a challenging process that can provide interesting results. Pachet (Pachet, 2003) uses a

Variable Length Markov Model for the "Continuator", an interactive application that analyses a player's input pattern of a MIDI keyboard and generates new sequences as soon as the player stops playing. Marchini (Marchini, 2010) also makes use of VLMMs for generating percussion sequences from a given example. Nayebi (Nayebi), exploits Long Short Term Memory Recurrent Neural Networks to compose music by using as an input a database of 20 sound files of various genres. In the next sections two of the main methods for predicting and generating new musical sequences that were investigated will be presented.

2.5.1 Markov Models

A Markov chain is a stochastic process that can predict a new event by considering only its past steps. A simple Markov process is demonstrated in Figure 7. If our current state is B, there is a 10% probability that our next state is still going to be in B, 50% probability that our next state is going to be A and 40% probability that our next state is going to be C. A first order Markov chain depends only the last step of a sequence, while an order-d will look at the length of d in order to predict a future event (Pachet F. &., 2011):

$$P(s_i|s_1,\cdots,s_{i-1}) = P(s_i|s_{i-d},\cdots,s_{i-1})$$

A great amount of thought has to be given while choosing a Markov order. A relatively low order can fail to imitate a desired sequence while a high order could possibly copy the whole sequence (Dubnov, 2003). A substantial drawback of Markov Models is lack of ability to emulate a pattern on a large scale (Pachet, 2003), although this is not an issue on this project since we are mainly interested in generating new pattern for a short period of time, and not emulating the whole structure of a musical piece.

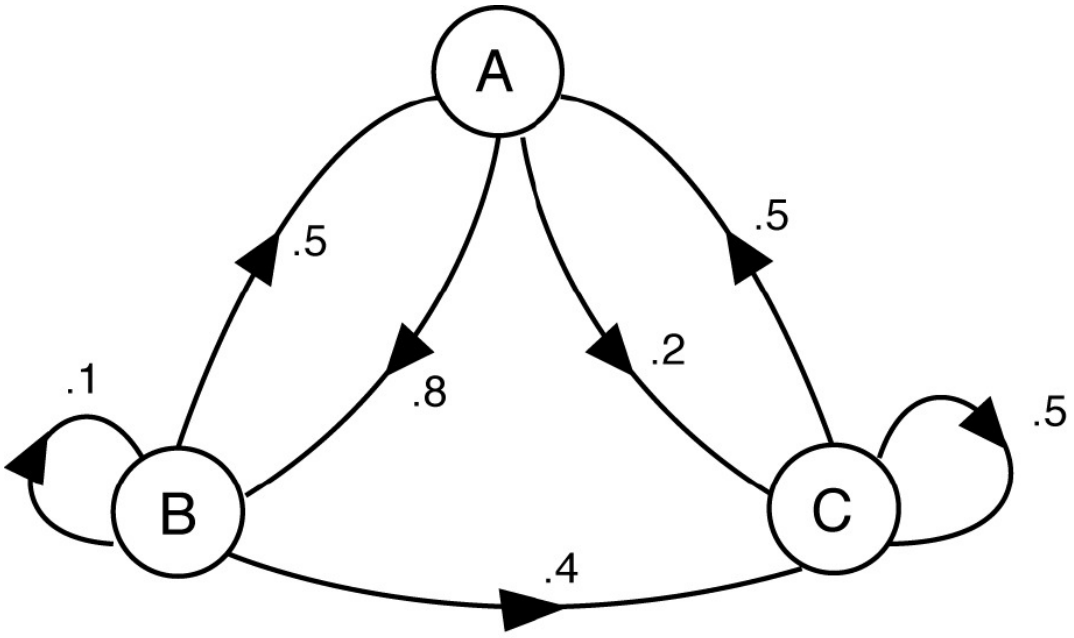

Figure 7 - A Markov Chain Example

Hidden Markov Models

Hidden Markov Models (HMM) are much more expressive than the simple Markov chains. As the name suggests, the states in HMMs are hidden, although we are able to calculate the probability on which state sequence resulted in the observations. By looking at a stock market example in Figure 8, we can see that we are no longer able to tell from which were we have moved to the next state, thus the states are hidden (Blunsom, 2004).

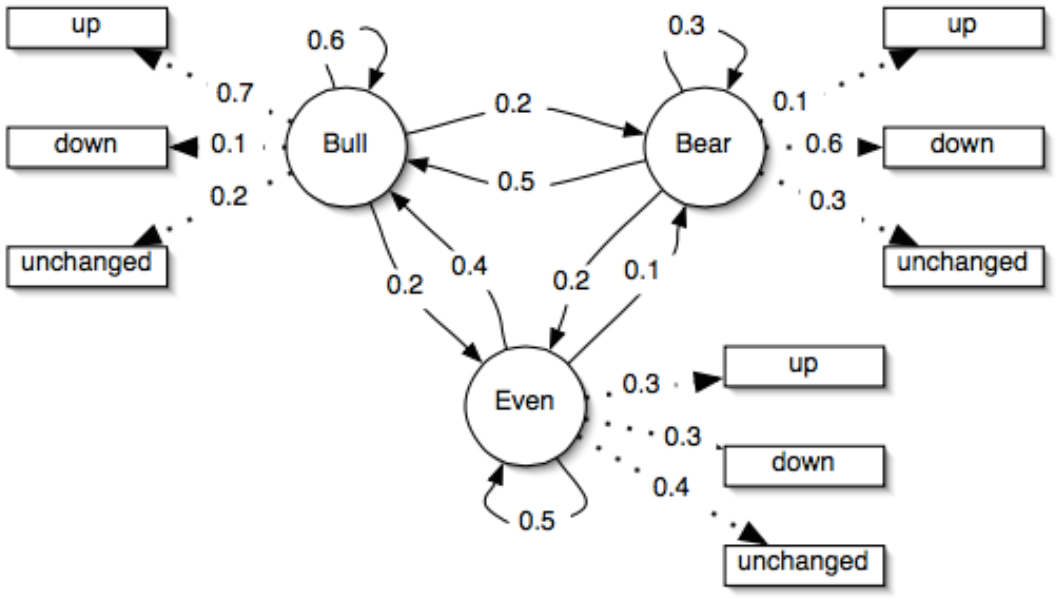

Figure 8 - A Hidden Markov Model example

Variable Length Markov Models

Variable Length Markov Models consider chains of varying orders and in some cases can prove to be more useful than HMMs since they are easier to work with than HMMs (Pachet F. &., 2011). The transition probabilities can be estimated by measuring how many times the states appear.

2.6 Open Sound Control Protocol

Open Sound Control (OSC) is a protocol for communicating between different platforms, and was developed by Wright and Freed in 1997 (Freed, 2009). It can be thought of as the successor of the MIDI (Musical Instrument Digital Interface) protocol. OSC uses an open ended URL style name and allows sending a single message to numerous receivers (Freed, 2009).

CHAPTER 3: IMPLEMENTATION

The implementation process is divided into two sections, the Timbre Analysis and Classification in Pure Data and the Variable Length Markov Chain sequence prediction in Python.

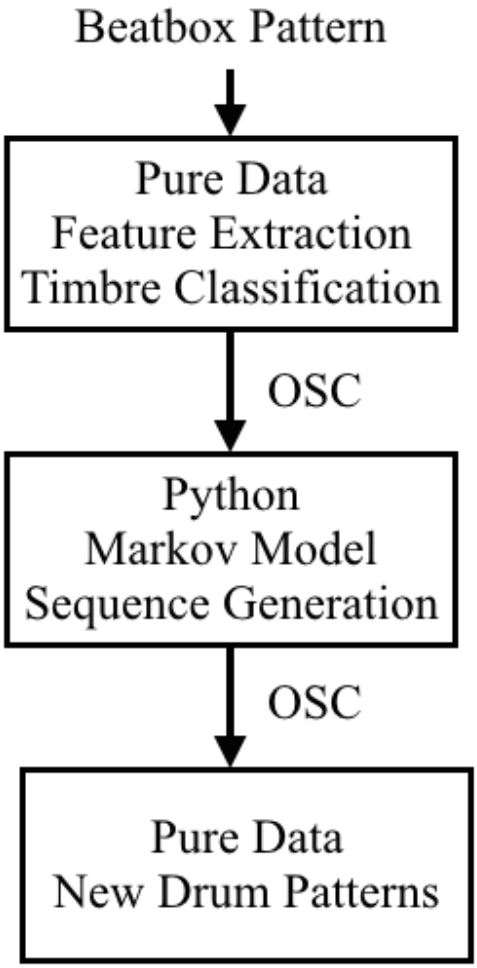

Figure 9 - Flow Diagram

3.1 Pure Data Implementation

The main tool that was used for the feature extraction and classification was the TimbreID Pure Data library by William Brent (Brent, A timbre analysis and classification toolkit for pure data, 2010) The first step of the process was the selection of the feature extraction object. The decision of using the bfcc~ was taken after a test comparison between the mfcc~ and the bcc~ objects. Bfcc~ appeared to provide slightly better results for the purpose of this project. A bfcc~ object was created with a window size of 1024 and a normalized Bark-spacing of 0.5. The bfcc~ object needs to be triggered using the "bang" object in order to output the BFCC features. A subpatch that includes bark~, one of the onset detection objects that were used on this project is connected to the inlet of the bfcc~. It is used for triggering the bfcc~. Bark~ is delayed by ½ of the analysis period that takes to fill the window(default). Connected to the subpatch that contains bark~, is the readf~ object, which is used for playing back the sound files of the training and testing.

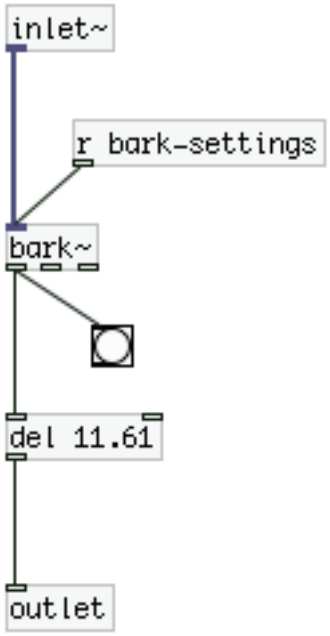

Figure 10 - Bark~, one of the onset detection objects

A list (split) object is connected on the outlet of the bfcc~ object. The list takes as an argument the number 12 in order to send 12 of the BFCC features to the timbreID classification object.

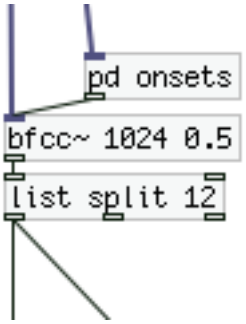

Figure 11 - The Bfcc~ object

A list (split) object is connected on the outlet of the bfcc~ object. The list takes as an argument the number 12 in order to send 12 of the BFCC features to the timbreID classification object.

TimbreID

On the selection of number of clusters the default number of clusters was kept (12) even though it is a large number, when using only three sounds. As with the majority of the objects in the timbreID library, the default values that were used appeared to work exceptionally well, once the appropriate sounds for training the trimbreID classifier were found. Additionally the Euclidan distance was chosen as opposed to Manhattan and Pearson Correlation Coefficient. Out of the left outlet of the timbreID object, the values that correspond to the trained drum sounds are outputted. They go through two different paths. One of them connected to the OSC objects, where they are packed and sent to Python. The other path of the outlet of the timbreID is responsible for triggering the synthetic drums. They first go through the spigot object, which acts as a gate and makes sure that they values trigger the synthetic drums only when the id toggle button is switched on. This way the drums are not triggered when the training process is happening.

Values that correspond to drums:

Kick drum	*Snare drum*	*Hi hat*
0 - 4	*5 - 9*	*10 - 14*

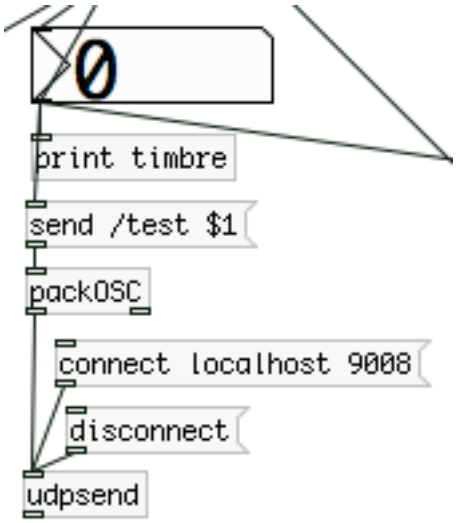

Figure 12 – OSC Client

When the values from the timbreID pass through the spigot gate object, go through the route objects. The route object makes sure the right drums get

triggered. For example if the number that exits the timbreID object is between 0 and 4, the route objects will trigger the kick drum.

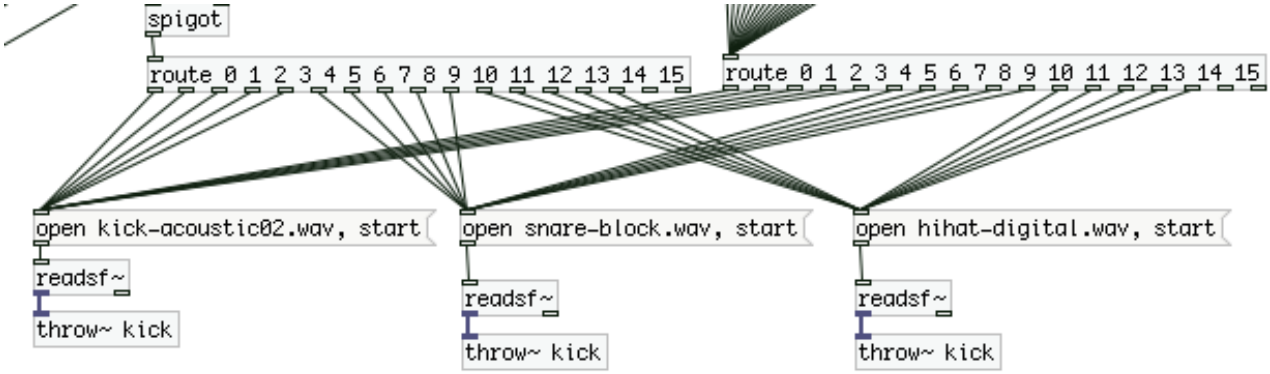

Figure 13 - The synthetic drums that get triggered from timbreID

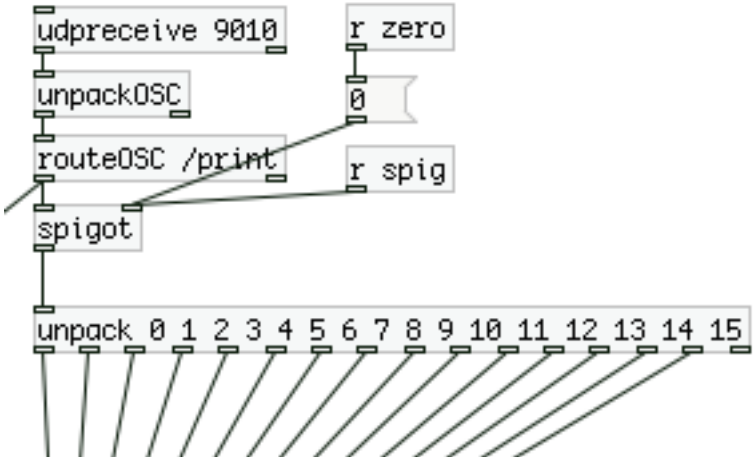

Figure 14 - OSC Server

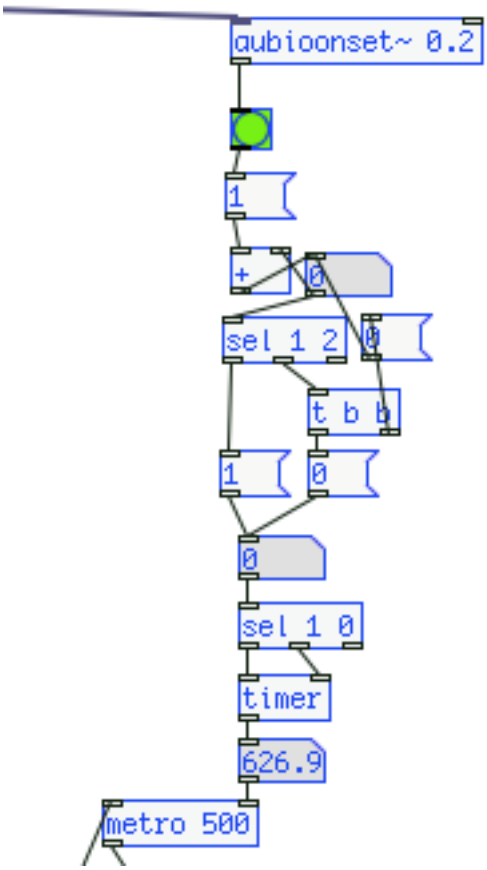

Figure 15 - Tap Tempo

Additionally a small mechanism was created that would act as a gate for the incoming OSC messages from Python. This mechanism was built so the generated new sequence that comes through the OSC server from Python doesn't trigger the synthetic drums while the user is using the system. The counter object was used for this purpose. The counter counts every bang it receives from its left inlet, which is connected to the tap tempo mechanism.

The moses object is also used for this purpose. Moses sends the numbers that are smaller from its argument on the left outlet and numbers that are bigger on its right outlet. The number seven was used, meaning that if the counter doesn't receive seven repeated bangs, the user is not using the system anymore and the OSC values from Python can start triggering the drums. Additionally wile the number of bangs is under seven the gate from the OSC server will be constantly closed.

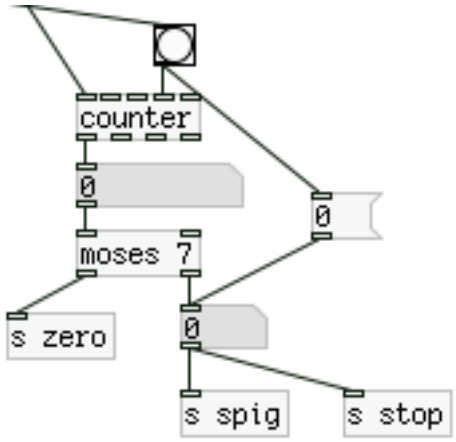

Figure 16 - Gate mechanism for the OSC server

3.2 Implementation in Python

The sequence prediction and generation was implemented in Python using a VMMM package from (c0z3n). A new Chain object is created that contains an empty chain. The addSequence method is used that takes as an argument a list containing the sequences values that arrive from PD via OSC.

Additionally the getSequence returns a sequence that is chosen from the Chain() class. A for loop is used for printing out the generated sequence. A global variable *lista* is created that will be storing all the incoming values from PD. After it has been declared as a variable, *lista* is assigned as an empty list. Inside the printing_handler function of the OSC package, the append method is used on the list, and all the values from the OSC server are now stored inside the *lista* list.

Most of the part of the code that is needed to run in real time is kept inside an infinite while loop which is running from an Exception. The loop stops only when the program is stopped. Inside this loop the sleep function with an argument of one second makes sure the while loop is not going to crash the computer. The *lista* list is mapped as a string so it can be used in the VMM addSequence method. The generated sequence is stored in the *out* variable, where it gets converted to an integer so it can be sent to PD via the OSC client. The append and send methods are used for sending the sequences to PD

CHAPTER 4: EVALUATION

The evaluation of the system was performed using the ENST drum loops (Gillet, 2006) and actual beatbox sequences. The bffc~ provided rather impressive results, with most of the sounds being classified correctly, when it was tested with the ENST drum loops. In some parts the open high hat was not classified correctly, however only a closed high hat was used in the training. In the case of an actual beatbox the performance was not that impressive, and instead of a kick drum a snare was mapped instead in some parts of an actual beatbox sequence.

On the table below, a comparison between the original and generated sequences from the files 039_phrase_disco_simple_medium_sticks and 040_phrase_disco_simple_fast_sticks

K = Kick drum

S = Snare

H = Closed Hi hat

Original	Generated		Original	Generated

K	**K**		**K**	**K**
H	**H**		**H**	**S**
K	**K**		**S**	**K**
H	**H**		**K**	**H**
S	**K**		**H**	**S**
H	**H**		**S**	**K**
K	**H**		**H**	**H**
H	**S**		**K**	**H**
K	**H**		**H**	**H**
H	**K**		**S**	**H**
S	**K**		**K**	**K**
H	**K**		**H**	**S**
K	**K**		**S**	**H**

H	**S**		**H**	**H**
S	**H**		**K**	**S**
H	**S**		**H**	**S**
K	**H**		**S**	**H**
K	**S**		**H**	**K**
H	**S**		**K**	**S**
S	**H**		**K**	**K**
H	**H**		**S**	**K**
K	**H**		**K**	**S**
H	**K**		**H**	**S**
S	**S**		**S**	**S**
H	**H**		**H**	**K**

CHAPTER 5: CONCLUSION AND FUTURE WORK

Overall a system that maps input beatbox sequences to synthetic drums as well as a sequence continuation method were implemented. The classification of the beatbox sequences provided satisfying results although certain issues need to be addressed. Firstly, the tap tempo mechanism doesn't function properly since it is triggered by every onset that gets detected from the onset detection object. Thus, a more sophisticated tempo detection mechanism needs to be implemented. The incoming prediction sequences that are generated fro the VMM are not mapped properly to the route object. Moreover the rhythmic structure of the input sequence is not preserved from generation of the new patterns coming from the VMM.

ACKNOWLEDGMENTS

I would like to thank my supervisor Hendrik Purwins for his assistance, support and supervision throughout this project.

**********2

BIBLIOGRAPHY[1]

Bello, J. P. (2005). *A tutorial on onset detection in music signals.* Speech and Audio Processing, IEEE Transactions on, 13(5), 1035-1047.

Blunsom, P. (2004). *Hidden markov models.* Lecture notes, August, 15, 18-19.

Brent, W. (2010). *A timbre analysis and classification toolkit for pure data.* Ann Arbor, MI: Michigan Publishing, University of Michigan Library.

Brent, W. (2009). *Perceptually based pitch scales in cepstral techniques for percussive timbre identification (pp. 3-6).* Ann Arbor, MI: Michigan Publishing, University of Michigan Library.

Duxbury, C. B. (2003). *Complex domain onset detection for musical signals.* In Proc. Digital Audio Effects Workshop (DAFx) (No. 1, pp. 6-9).

Dubnov, S. A. (2003). *Using machine-learning methods for musical style modeling.* Computer, 36(10), 73-80.

Dave, N. (2013). *Feature extraction methods LPC, PLP and MFCC in speech recognition.* International Journal for Advance Research in Engineering and Technology.

Freed, A. &. (2009). *Features and Future of Open Sound Control version 1.1 for NIME.* In NIME (Vol. 4, No. 06, p. 2009).

Gulzar, T. S. (2014). *Comparative analysis of LPCC, MFCC and BFCC for the recognition of Hindi words using artificial neural networks.* International Journal of Computer Applications.

Gillet, O. &. (2006). *ENST-Drums: an extensive audio-visual database for drum signals processing.* In ISMIR (pp. 156-159).

Imai, S. (1983). *Cepstral analysis synthesis on the mel frequency scale.* Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'83. (Vol. 8, pp. 93-96). IEEE.

Hazan, A. (2005). *Performing expressive rhythms with billaboop voice-driven drum generator.* In Proc. of the 8th Int. Conference on Digital Audio Effects.

Hasan, M. R. (2004). *Speaker identification using mel frequency cepstral coefficients.* variations, 1, 4.

Janer, J. (2005). *Feature extraction for voice-driven synthesis.* In Audio Engineering Society Convention 118. Audio Engineering Society.

Logan, B. (2000). *Mel Frequency Cepstral Coefficients for Music Modeling.* ISMIR.

Nayebi, A. &. *GRUV: Algorithmic Music Generation using Recurrent Neural Networks.*

Marchini, M. (2010). *Unsupervised generation of percussion sequences from a sound example.* Master's thesis.

Mehta, L. R. (2013). *Comparative study of MFCC and LPC for Marathi isolated word recognition system.* Int J Adv Res Electr Electr Instrum Eng.

Pachet, F. &. (2011). *Markov constraints: steerable generation of Markov sequences. Constraints.* Constraints, 16(2), 148-172.

Pachet, F. (2003). *The continuator: Musical interaction with style.* Journal of New Music Research, 32(3), 333-341.

Sinyor, E. R. (2005). *Beatbox classification using ACE.* In Proceedings of the International Conference on Music Information Retrieval.

Shiffman, D. S. (2012). *The nature of code.* D. Shiffman.

Shrawankar, U. &. (2013). *Techniques for feature extraction in speech recognition system: A comparative study.* arXiv preprint arXiv:1305.1145.

Stevens, S. S. (1937). *A scale for the measurement of the psychological magnitude pitch.* The Journal of the Acoustical Society of America.

Stowell, D. a. (2010). *Delayed decision-making in real-time beatbox percussion classification.* Journal of New Music Research 39.3 (2010): 203-213.

Stowell, D. (2010). *Making music through real-time voice timbre analysis: machine learning and timbral control.* Queen Mary, University of London.

Randall, R. B. (1981). *Cepstrum Analysis, Technical Review.* Brüel & Kjær.
Randall, R. B. (2012). *New cepstral methods of signal preprocessing for operational modal analysis.* In Proc. Int. Conference on Noise and Vibration Engineering (ISMA).

**********2