# An algorithm to fade electronic rhythms together

Siddharth Verma

15th June 2016

## 1 Abstract

Beatmasher is an Ableton Live MAX/MSP and Python program created to mix two electronic rhythms. It is different from traditional DJ software as it fades notes, not volume. This paper delineates the algorithm Beatmasher uses to remove notes from a rhythm while keeping its character.

## 2 Background

A track is a pattern of hits on a single drum or sample. It is represented as a list of 0s and 1s. For example, the standard quarter note kick drum pattern prevalent in all electronic music can be represented as $\vec{t}$ in equation 1.

$$\vec{t} = \langle 1, 1, 1, 1 \rangle \tag{1}$$

In equation 1, a 1 stands for a note on, or a hit on the kick drum. One bar is the entire length of the pattern. To play this bar, the program reads from left to right, plays the kick when it reaches a 1, and loops to the beginning when it reaches the end of the bar. Each 0 or 1 lasts for a fixed duration and this duration is the resolution of the track. The resolution of the above track is four, but the resolution chosen for BeatMasher was thirty-two notes in a bar. So, the same pattern would be represented in the program as $\vec{t}$ in equation 2:

$$\begin{aligned} \vec{t} = \quad & \langle 1, 0, 0, 0, 0, 0, 0, 0, \\ & 1, 0, 0, 0, 0, 0, 0, 0, \\ & 1, 0, 0, 0, 0, 0, 0, 0, \\ & 1, 0, 0, 0, 0, 0, 0, 0 \rangle \end{aligned} \tag{2}$$

The zeros tell the program to do nothing for that duration, A 1 is called a note, and a 0 is called a rest.

A beat is a collection of these tracks for different percussion items. It represents the entire electronic rhythm that is to be played. For example, a beat could contain a hi-hat track, a kick track and a snare track.

## 3 Introduction

The traditional method of DJing is to fade two electronic rhythms into each other by modulating volume. However, this method is not effective when multiple rhythms are to be mixed as the sound gets easily cluttered. The better approach is to make them rarer; to remove notes so that the rhythm becomes sparser while maintaining the same quality or character.

The aim of this program is to have a slider that can progressively remove notes to make rhythms sparser in order to mix electronic rhythms together. This is a complicated problem as making a decision as to which notes to take out is a complex musical decision.

This problem was solved by assuming that the notes which should be retained are the most unique ones. This assumption can be made because it is the rarest notes that separate a track from others. Removing the most common notes allows multiple rhythms to be overlaid with minimal overlap.

To find the most unique parts of a beat, the most unique track and the most unique notes in a track must both be found. Then, notes can be taken out of tracks in the order of uniqueness of tracks and of notes.

## 4 Track Uniqueness

To decide the order of importance of different tracks, four criteria were used - periodicity, dissimilarity, number of notes, and user defined uniqueness. Each of these was given a value between zero and one. They were then weighted and averaged to give a final value of uniqueness between zero and one, which was used to compare it with different tracks.

### 4.1 Periodicity

The periodicity of a track is the length after which it repeats. If this length is long, then it signifies that the track doesn't repeat often and is therefore unique. Periodicity was measured by incrementally phase shifting the track and performing an XOR with the original track. The periodicity is the shortest phase shift for which the sum of all the numbers of the resultant track is zero.

For example, let us consider a track $\langle 1, 0, 0, 1, 0, 0 \rangle$. This track has a periodicity of three, and is found by the following method - The track is shifted by incremental values, and the following operation is done:

offset = 1

$$\begin{aligned} \vec{t_1} &= \langle 1,0,0,1,0,0 \rangle \\ \vec{t_2} &= \langle 0,1,0,0,1,0 \rangle \\ \vec{t_1}\ XOR\ \vec{t_2} &= \langle 1,1,0,1,1,0 \rangle \\ \sum(\vec{t_1}\ XOR\ \vec{t_2}) &= 4 \end{aligned} \tag{3}$$

offset = 2

$$\begin{aligned} \vec{t_1} &= \langle 1,0,0,1,0,0 \rangle \\ \vec{t_2} &= \langle 0,0,1,0,0,1 \rangle \\ \vec{t_1}\ XOR\ \vec{t_2} &= \langle 1,0,1,1,0,1 \rangle \\ \sum(\vec{t_1}\ XOR\ \vec{t_2}) &= 4 \end{aligned} \tag{4}$$

offset = 3

$$\begin{aligned} \vec{t_1} &= \langle 1,0,0,1,0,0 \rangle \\ \vec{t_2} &= \langle 1,0,0,1,0,0 \rangle \\ \vec{t_1}\ XOR\ \vec{t_2} &= \langle 0,0,0,0,0,0 \rangle \\ \sum(\vec{t_1}\ XOR\ \vec{t_2}) &= 0 \end{aligned} \tag{5}$$

For an offset of one in equation 3 and two in equation 4, the sum is four, but for an offset of three in equation 5 the sum is zero. Thus, the periodicity $P = 3$. The maximum periodicity possible is the length of the track.

### 4.2 Dissimilarity

For a track to be unique, it must be dissimilar from other tracks. To find the dissimilarity between two tracks, one can sum the XOR of the two tracks divided by the length of the track, as shown below:

Take two example tracks $\vec{t_1} = \langle 1,0,0,0,1,0,0,0 \rangle$ and $\vec{t_2} = \langle 0,1,0,0,1,0,1,1 \rangle$. The dissimilarity index $D$ of these two can be calculated as shown in equation 6.

$$\begin{aligned} D &= \frac{\sum \vec{t_1}\ XOR\ \vec{t_2}}{len(\vec{t_1}\ XOR\ \vec{t_2}} \\ &= \frac{\sum \langle 1,1,0,0,0,0,1,1 \rangle}{len(\vec{t_1}\ XOR\ \vec{t_2})} \\ &= \frac{4}{8} = 0.5 \end{aligned} \tag{6}$$

This process was done for each track against a set of tracks in a database of common electronic beats, and the indices were averaged to get a final index of dissimilarity.

### 4.3 Number of notes in track.

The assumption that a track that contains less notes is unique can be made, and the uniqueness is calculated by equation 7.

$$N = \frac{number\ of\ notes}{length\ of\ track} \tag{7}$$

### 4.4 Uniqueness given by user.

Finally, a uniqueness $R$ is given by the user as all the above factors do not account for the sound design of the instrument; a rare dubstep growl is more unique than a banal kick.

### 4.5 Total uniqueness

Using constants $k_P$, $k_D$, $k_N$ and $k_R$, the final uniqueness $U$ is the weighted average show in equation 8.

$$U = \frac{k_P P + k_D D + k_N N + k_R R}{k_P + k_D + k_N + k_R} \tag{8}$$

## 5 Note uniqueness

The problem of note uniqueness at first seemed, as a rule based system would require a lot of convoluted rules. However, using a data driven approach simplified the problem vastly.

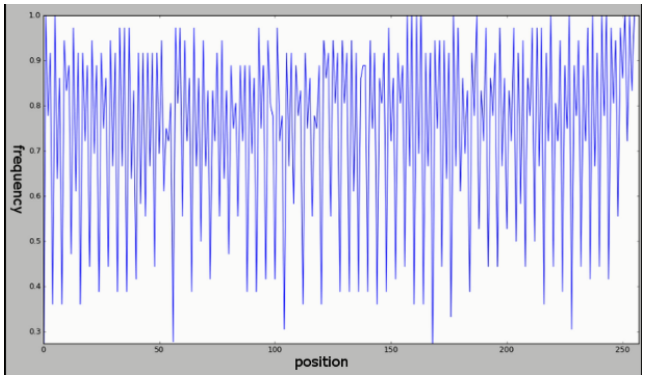


Figure 1: Position vs Relative Frequency of Notes

Figure 1 is a normalized count of how many times a note appears at a particular position in a database of standard electronic beats. Since the opposite of being unique is to appear many times, an inversion of this graph gives position versus uniqueness of notes. This process was run against all tracks in the database of standard electronic beats.

To get the resultant uniqueness given the position of a note in a bar, one just needs to look up the y value for a given x value in the graph. This value was used to compare notes and decide which ones to remove first.

## 6 Conclusion

This program was developed at the Workshop in Algorithmic Computer Music under professor David Cope. Although it works well, it doesn't work perfectly. The algorithm only works on electronic music, and doesn't take into account the context, the tracks into which the selected one will be overlaid. Occasionally, This causes minor clashes when two notes are played at the same time. Furthermore, sometimes the notes fade out too quickly or slowly due to the nature of the algorithm.

A video containing an explanation and demonstration of the program can be found at `https://www.youtube.com/watch?v=KfH17ruxRkY`