USING THE FIBONACCI SEQUENCE TO TIME-STRETCH RECORDED MUSIC

David Su

MIT Media Lab
75 Amherst St
Cambridge, MA
davidsu@mit.edu

ABSTRACT

Human creativity in the music production process is often augmented by technological tools that modify existing musical input, thereby generating unanticipated ideas. The Fibonacci stretch algorithm is a means of time-stretching a music recording, using the Fibonacci sequence as a theoretical basis, such that the resulting music recording is perceived as being in a new meter. Because the overall impulse shape of the recording remains intact, Fibonacci stretch generates novel rhythmic ideas rooted in a natural-sounding transformation of existing musical material. This paper explores the possibilities for Fibonacci stretch to assist in the music production process, and also introduces an audio plugin prototype that encompasses a real-time variant of the algorithm.

1. INTRODUCTION

At the core of music production, as with any creative endeavor, is the generation of ideas. In music production this generation often comes in the form of *modification* of existing musical material via technology. The application of audio effects can be seen as a method of automating digital signal processes that allow a producer to explore and iterate upon ideas; indeed, the term "audio effect" implies alteration rather than generation from scratch.

Fibonacci stretch describes a method by which time-stretching procedures are applied, in conjunction with rhythmic principles based upon the Fibonacci sequence, to an existing audio track. As an audio effect, Fibonacci stretch can be applied both in an offline setting and in a real-time context, yielding unconventional results that nevertheless can be perceived as natural rhythmic modifications of the original music.

2. FIBONACCI RHYTHMS AND SCALING

The theoretical basis for Fibonacci stretch draws from rhythm perception research, in particular Iyer's work with rhythmic expansion and contraction along the Fibonacci sequence [1] and Toussaint's research on Euclidean rhythms [2].

Figure 1 shows various representations of the *tresillo* rhythm [3]. If the p refers to the number of impulses (or onsets) in the rhythm, and s refers to the number of steps (or subdivisions) in the rhythm, we can describe this rhythm as having $p = 3$ and $s = 8$. In its binary sequence representation $[1, 0, 0, 1, 0, 0, 1, 0]$ it falls within the family of "binary sequences generated by Bjorklund's algorithm" [2], and thus can be classified as a Euclidean rhythm. Specifically, the rhythm can be generated by supplying $n = p$ and $m = s$ to Bjorklund's algorithm, which "solves the general problem of distributing n impulses over m 'timing slots' in the most even way possible" [4].

We define a Fibonacci rhythm as a rhythm whose impulse lengths are all Fibonacci numbers. The *tresillo* rhythm can also be classified as a Fibonacci rhythm, as the lengths of its impulses are $[3, 3, 2]$. We can expand this rhythm along the Fibonacci sequence by replacing each impulse length with the Fibonacci number that follows it, yielding $[5, 5, 3]$. Applying this procedure twice in a row yields $[8, 8, 5]$. Likewise, we can contract the *tresillo* rhythm by replacing each impulse length with the preceding Fibonacci number, yielding $[2, 2, 1]$.

This manipulation of Fibonacci rhythms can be expressed in terms of an integer *scale factor*. Positive scale factors correspond to rhythmic expansions and negative scale factors correspond to contractions, while the scale factor's magnitude representing how many times to apply the scaling procedure along the Fibonacci sequence. For example, scaling the *tresillo* rhythm by a factor of 2 yields the rhythm $[1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0]$, while scaling it by a factor of -1 yields $[1, 0, 1, 0, 1]$.

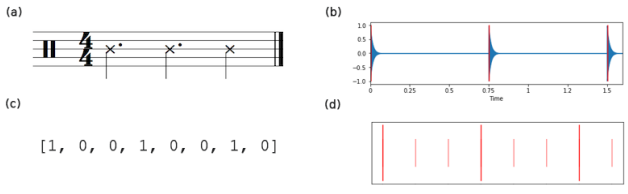

Figure 1: The *tresillo* rhythm: **(a)** as Western notation, **(b)** as an amplitude envelope of clicks, **(c)** as a binary sequence, **(d)** as a series of vertical lines denoting impulses and steps.

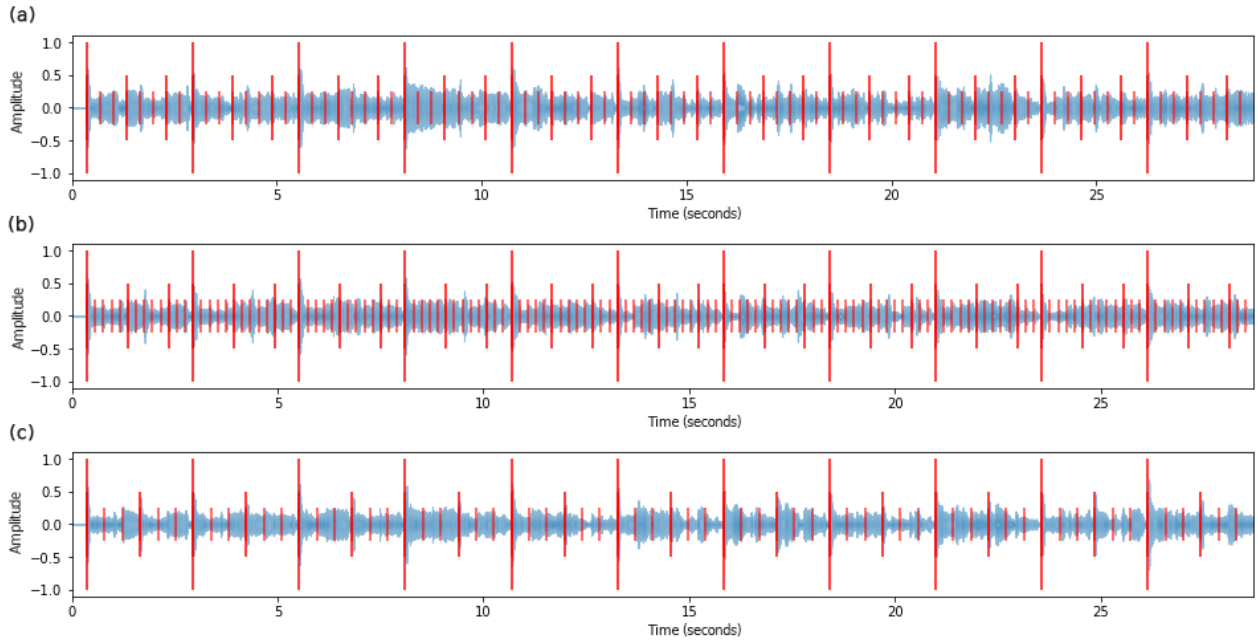

Figure 2: The first 30 seconds of Michael Jackson's "Human Nature": **(a)** in its original form with the *tresillo* rhythm (which corresponds to the rhythm of the bassline) overlayed in red, **(b)** after applying Fibonacci stretch using $r = [1, 0, 0, 1, 0, 0, 1, 0]$ and $f = 1$ with the target rhythm overlayed in red, **(c)** using $r = [1, 0, 1, 0, 1, 0, 1, 0]$ and $t = [1, 0, 0, 1, 0, 0]$, essentially converting from 4/4 to 6/8 meter. Audio examples can be found at `http://usdivad.com/fibonaccistretch/examples.html`.

3. THE FIBONACCI STRETCH ALGORITHM

Fibonacci stretch works by applying the aforementioned scaling procedure to audio recordings. In its basic form it operates on the level of musical measures, although it can be modified to accomodate other rhythmic levels such as the *tatum* or *tactus* level [5]. The audio recording, an original rhythm r, and a scale factor f are given as input. The Fibonacci stretch procedure then time-stretches each rhythmic impulse of the audio recording as well as the impulse's enclosed steps. The modified audio data are then concatenated and returned; time-stretching is performed such that the final length of the modified audio recording is the same as that of the original recording.

The scale factor f can be considered a shorthand for generating a target rhythm t, such that t is a Fibonacci rhythm for which each impulse length is f Fibonacci numbers away from its original impulse length in r. A specific target rhythm t can be supplied in lieu of f in order to accomodate non-Fibonacci rhythms. While this can lead to results that are divorced from the mathematical properties of the Fibonacci sequence, it does allow for a larger output space.

The rate at which each impulse is time-stretched is derived from the ratios between the lengths of corresponding impulses in r and t. For example, if $r = [1, 0, 0, 1, 0, 0, 1, 0]$ and $t = [1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0]$, then the three impulses in r are time-stretched by 0.6, 0.6, and $0.\dot{6}$ respectively (with dots above digits representing recurring decimals). In addition, impulses are further subdivided by generating Euclidean rhythms in order to stretch the interspersed steps in a perceptible manner. If r_0 and t_0 are the first impulses of r and t respectively, then $pr = [1, 0, 0]$ and $pt = [1, 0, 0, 0, 0]$. To calculate the stretch rate of each step in r_0, a Euclidean rhythm c is generated such that its number of steps is equal to the number of steps in t_0 (in this case, 5) and its number of impulses is equal to the number of *steps* (not impulses) of r_0 (in this case, 3); thus, $c = [1, 0, 1, 0, 1]$. Ratios can then be calculated between each step in r_0 and its corresponding impulse in c. In this case, each step in r_0 would be further time-stretched by 0.5, 0.5, and 1.0 respectively. The resulting time-stretch rates s, such that each step at index i of r is time-stretched by the rate at index i of s, can be calculated by multiplying the step and impulse time-stretch rates together. In this case, $s = [0.3, 0.3, 0.6, 0.3, 0.3, 0.6, 0.\dot{6}, 0.\dot{3}]$.

Because the ratio of successive values in the Fibonacci sequence converges to ϕ, the overall impulse shape of the recording remains intact. As such, if both r and t are Fibonacci rhythms, then the resulting transformations can be perceived as natural rhythmic expansions or contractions of the original music.

The Fibonacci stretch procedure is agnostic to specific time-stretching algorithms. In its current implementation,

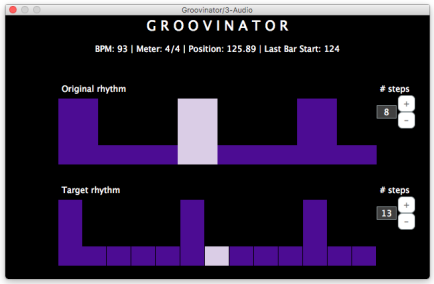

Figure 3: The Groovinator plugin, which applies a variant of Fibonacci stretch in real-time.

sample-level time-stretching was performed using the librosa package [6], which stretches audio via a phase vocoder based on [7]. The division of steps is performed by simply dividing each measure into s segments of equal length; currently, no onset detection or beat tracking is performed.

Figure 2 depicts the results of applying Fibonacci stretch, with various input parameters, to the first 30 seconds of Michael Jackson's "Human Nature" [8].

4. APPLICATIONS TO MUSIC PRODUCTION

The musically interesting results of applying Fibonacci stretch to example recordings led to the development of a prototype audio plugin called the Groovinator. The primary purpose for the development of this plugin was to evaluate the effectiveness of Fibonacci stretch in a real-time context. Such a context introduces several constraints, the most significant one being that because samples are received in real-time, a sample cannot be processed for output before it is encountered as input. As a result, real-time rhythmic contraction cannot be achieved unless the step containing the designated input appears before the time at which the stretched audio is to be output; this limits the output space of possible target rhythms. The Groovinator plugin, shown in Figure 3, was developed using the JUCE framework and is available in VST and AU format.

5. FURTHER WORK

There are a number of further avenues to explore as well as implementation improvements to consider that would have an impact on both the offline and real-time versions of Fibonacci stretch. For example, the quality of the resulting audio could be improved by more intelligent onset detection, such that the time-stretching procedure stretches tails more than onsets. The balance between user control and intelligent behavior poses interesting challenges on that front. Similarly, tempo detection, as well as the ability to vary tempo over time, would allow for more musical expressivity.

In addition, because the Fibonacci sequence itself grows at an exponential rate, it is often the case that using scale factors above a certain value leads to virtually imperceptible differences; the *tresillo* rhythm stretched by a factor of 6 yields a target rhythm with 144 steps, which becomes difficult to perceive when given the same duration as the original 8 steps. Indeed, most of the musically satisfying results arise from applying the procedure with a scale factor between -1 and 3. Exploring this range of results could yield further insight; this would necessitate a more rigorous treatment of the perceptual qualities of the Fibonacci sequence and the golden ratio when applied to rhythm.

6. CONCLUSION

The Fibonacci stretch algorithm produces novel musical ideas via sample-level manipulation of audio that can be difficult and time-consuming to recreate by hand. In this way it opens up possibilities for rapid experimentation of rhythm in a music production setting. Implementations and examples of the algorithm as well as the audio plugin are available on GitHub [1]. Any feedback, contributions, or musical usage are highly welcomed and encouraged.

7. REFERENCES

[1] V. Iyer, "Strength in numbers: How Fibonacci taught us how to swing," *The Guardian*, 2009.

[2] G. Toussaint, "The Euclidean algorithm generates traditional musical rhythms," in *Proceedings of BRIDGES: Mathematical Connections in Art, Music and Science*, pp. 47–56, 2005.

[3] D. Peñalosa and P. Greenwood, *The Clave Matrix: Afro-Cuban Rhythm: Its Principles and African Origins*. Bembe Books, 2009.

[4] E. Bjorklund, "The theory of rep-rate pattern generation in the SNS timing system," *SNS ASD Tech Note, SNS-NOTE-CNTRL-99*, 2003.

[5] A. P. Klapuri, A. J. Eronen, and J. T. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 342–355, 2006.

[6] B. McFee, M. McVicar, C. Raffel, D. Liang, O. Nieto, J. Moore, D. Ellis, D. Repetto, P. Viktorin, and J. F. Santos, "Librosa: v0. 4.0," 2015.

[7] D. P. W. Ellis, "A phase vocoder in Matlab," 2002. Web resource.

[8] M. Jackson, *Thriller*. Epic, 1982.

[1] `https://github.com/usdivad/fibonaccistretch`