DESIGN OF TIMBRE WITH CELLULAR AUTOMATA AND *B*-SPLINE INTERPOLATION

Matthew Klassen
DigiPen Institute of Technology, Redmond, WA
mattjklassen@gmail.com

Paul Lanthier
Institut polytechnique Unilasalle
Mont Saint Aignan, France
planthier76@outlook.fr

ABSTRACT

The origin of this paper comes from the collaboration of the authors on the UPISketch software project (see [1]), which is a creation of the Iannis Xenakis Center and a descendant of the UPIC project of Xenakis. The software, created in 2017, allows the user to draw curves which can be interpreted as musical gestures, acting on elements such as pitch, time, and timbre. Two fundamental mathematical tools used in this process are splines, to model graphical gestures, and cellular automata, to generate musical gestures such as pitch sequences. In this paper we apply both of these techniques on the "micro-level" to the *timbre* of waveforms, using the approach of *cycle interpolation*. A waveform is modeled as a sequence of cycles, and each cycle is modeled as a cubic spline curve. The *B*-spline coefficients of each cycle form a discrete representation, which can be manipulated through the use of cellular automata. In this way, key cycles can be generated, and can be interpolated with *B*-splines to form new and interesting timbres. We illustrate this generation and design of waveforms in our own software implementation with JUCE, which in turn will become part of UPISketch.

1. UPISKETCH AND OUR MOTIVATION

In the UPISketch software, sound events are considered as sequences of musical elements such as: time duration, loudness or amplitude, pitch or fundamental frequency, and timbre or waveform. The space of configurations is thus a set of sequences over an alphabet inspired by those parameters. It is straight-forward to give numerical values to the first two parameters, duration and loudness. The pitch can also be represented in discrete segments, such as note values where the pitch is constant, or as continuous curves representing a glissando or other pitch variation. The notion of timbre, however, is recognized as being much more complicated. Perception of timbre can be modeled as multidimensional (see [2] and [3]), and its analysis can be approached in both time and frequency. In [2] the global time-envelope as well as spectral parameters such as spectral centroid are described in detail. We will consider some of these when we discuss our models involving splines and cellular automata. One approach to timbre, employed in many music software systems, is to assign a discrete collection of instrument sounds as available timbres. Another approach is to use synthesis methods, such as frequency modulation, to allow timbres to change continuously according to various parameters.

Our approach to timbre is both discrete and continuous. We model approximately periodic waveforms in terms of cycles, where each cycle is given a discrete representation as a set of cubic *B*-spline coefficients. If a sound is to have a timbre which is somewhat similar to an acoustic instrument, then this should be achievable by a gradual change in time of the shape of cycles, as can be observed in digital instrument sample recordings. A sequence of cycles can be extracted from a recorded sound, or can be generated by artificial means. The method of generation we consider in this paper uses cellular automata, allowing for discrete local changes in the *B*-spline basis coefficients which in turn can be smoothed, or interpolated, to create more gradual changes.

2. SPLINE INTERPOLATION

It is worth noting that the choice of modeling signals with cubic splines, or piecewise polynomials, has a basic effect on the timbre due to their inherent spectrum. In fact, interpolation with cubic splines can be thought of as a type of additive synthesis of periodic waveforms which are derived from the square wave. It is well known that the square wave has Fourier series with harmonic spectral decay $1/n$, and its integral the triangle wave, has spectral decay $1/n^2$. (See for example [4] chapter 7.) Repeating this process we obtain quadratic and cubic periodic waves, with spectral decays $1/n^3$ and $1/n^4$. Approximation of an audio signal with interpolating cubic splines inherits this type of spectrum, in addition to the modeling of lower frequencies, such as the fundamental, through the shaping of cycles.

We model short samples of an audio waveform at the level of one cycle, or period, although we do not impose the condition that our waveforms are strictly periodic. Further, we choose to represent each cycle as a C^2 interpolating cubic spline on the interval $[0,1]$ given in terms of a *B*-spline basis. Each cycle will be represented by a spline function $f(t)$ with values in the interval $[-1,1]$. We will also assume that the interval $[0,1]$ is evenly partitioned into k subintervals and that f is a cubic polynomial on each subintervals. For simplicity, each cycle will be assumed to start and end with value 0. It is then natural to use the knot sequence

$$\mathbf{t} = \{t_0, \ldots, t_N\} = \{0,0,0,0,\frac{1}{k},\frac{2}{k},\ldots,\frac{k-1}{k},1,1,1,1\}$$

with associated B-spline basis:

$$\{\mathscr{B}_0(t), \mathscr{B}_1(t), \ldots, \mathscr{B}_{n-1}(t)\}.$$

Each cubic B-spline basis function $\mathscr{B}_i(t)$ uses 5 consecutive knot values: $t_i, \ldots, t_{i+4}$ and is positive on the interval of its support: (t_i, t_{i+4}). The dimension is $n = k+3$ and thus $N = k+6$. In this way, the spline

$$f(t) = c_0\mathscr{B}_0(t) + \cdots + c_{n-1}\mathscr{B}_{n-1}(t)$$

will satisfy $f(0) = 0 = c_0$ and $f(1) = 0 = c_{n-1}$. The remaining B-spline coefficients can be considered a discrete representation of the cycle which can evolve through various discrete transformations. In some cases, cycle data may be derived from actual recorded sounds, or audio files, or cycles may be generated through algorithmic processes, or synthesis. The spline function f can be evaluated by nested linear interpolation, or the de Boor Algorithm:

de Boor Algorithm for B-spline evaluation:

First, set $c_i^0 = c_i$ for $i = 0, \ldots, n-1$. Next for $t \in [0,1]$ choose J so that $t \in [t_J, t_{J+1})$. Then for $p = 1,2,3$, for $i = J-3+p, \ldots, J$, set:

$$c_i^p = \frac{t - t_i}{t_{i+3-(p-1)} - t_i} c_i^{p-1} + \frac{t_{i+3-(p-1)} - t}{t_{i+3-(p-1)} - t_i} c_{i-1}^{p-1}$$

Finally, the output is $f(t) = c_J^3$.

The use of B-splines for audio waveform synthesis occurs in the work of Nick Collins [5], in which one cycle of the waveform can be manipulated by the movement of control points, as well as in the modeling of envelopes for F_0 synthesis as in [6]. Collins' software implementation provides the user with the ability to modulate the timbre of the resulting waveform by moving control points. Our approach is to replace the control by the user of individual cycles with cellular automata for the generation of key cycles. These key cycles are then interpolated to create "in between" cycles with B-spline interpolation. This extends the work in [7], where all key cycles are assumed to come from an audio signal. As initial input to the CA, we may also use one cycle from an original audio sample.

Since we may derive a cycle from actual audio data, we state a few more parameters which determine the model. It is useful to keep the number of subintervals k constant throughout a generated sequence of cycles, and to keep the subintervals of $[0,1]$ of uniform length. Then the dimension of the vector space V of C^2 cubic splines on the sequence of subintervals is $n = k+3$, with basis given as above. In order to approximate one cycle of a recorded audio waveform, we need to have n inputs in the interval $[0,1]$, which are evenly spaced, according to the Schoenberg Whitney Theorem on spline interpolation (see [8]). So we use the $k-1$ knot values which occur strictly inside the interval $[0,1]$, as well as two more values at the midpoints of the outer two subintervals: $\frac{1}{2k}$ and $1 - \frac{1}{2k}$. These $k+1$ inputs, together with the end conditions $f(0) = 0 = f(1)$ will then guarantee a unique solution in V. In figure 1 is an example of interpolating cubic spline to one cycle of audio data with 330 sample points. The spline, drawn here in blue, uses $n = 60$ interpolation points.

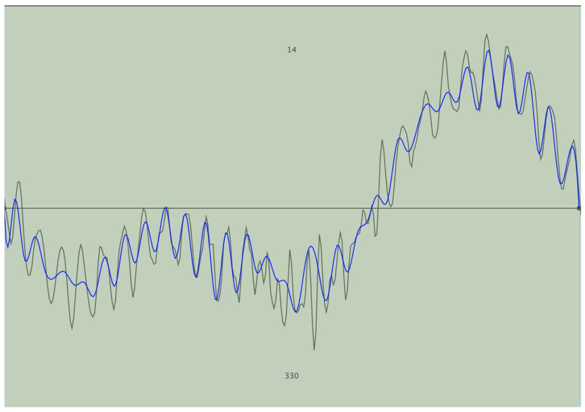


Figure 1. Spline interpolation of audio data

An analogy for this type of sound model comes from animation with key framing. Here, the cycle plays the role of one frame in animation, where the key frames might come from an artist's imaginative drawing, or they might come from motion capture data. In the same way, one key cycle might come from some idea for the design of timbre, or it might come from one cycle which is captured from a particular audio recording. In the previous paper [7] we explore mostly the latter case, but here we explore the former case: the design of timbre.

3. CYCLE INTERPOLATION (CI)

As in [7], we use a second set of splines to control the evolution of the cycles between key cycles. We call this set of splines *meta-splines* and the set of splines which model individual cycles *cycle-splines*. It is reasonable to use the same types for both sets, for instance C^2 cubic splines. However, for computational speed, especially when there are many key cycles to interpolate, we may also use piecewise linear meta-splines.

An important point, raised by Collins in [5], is that linear interpolation between spline coefficients can be equivalent to an "audio crossfade", meaning that the final samples produced are nothing more than the fade out of one block of samples while another is faded in. This is a consequence of the simple observation that a linear combination of B-spline coefficients produces an equivalent linear combination of output values of the spline functions. One way to avoid this type of crossfade is to supply cycles with an envelope, which can be as simple as a sequence of scaling factors for each cycle. Such scaling factors can be derived from realistic envelopes of actual recorded sounds, or generated in other ways. Any nonlinear choice of such scale factors will avoid the case that the resulting cycle interpolation becomes an audio crossfade.

Our synthesis model consists of a sequence of cycles C_j, some of which are key cycles and are generated by some algorithmic process, and the rest of which are interpolated. There are two main types of key cycle sequences: regular and irregular. The regular case places key cycles at regular time intervals, with a fixed number of intermediate cycles. The irregular case usually places more key cycles near the beginning of an audio sequence or waveform, in order to simulate an instrument sample which begins with an attack and decay phase. Such irregular sequences of positive integers can be generated in various ways, for instance with the classical Fibonacci sequence, or squares, etc.

4. CELLULAR AUTOMATA (CA)

A natural process in music composition is to develop interesting material from a simple element, such as the subject for a fugue, or a melody or phrase. The idea is to generate new elements through musical transformations, such as inversion and transposition. This can be extended further by using other properties, or musical dimensions, such as rhythm, pitch, dynamics, and also timbre. Such transformations of a musical element can be done locally, making changes to a note based on its neighbors for instance. This leads naturally to Cellular Automata (CA), which are computed by a local rule.

CA are already present in the UPISketch software, acting on parameters of pitch and time, as explained in [9], so it is compelling to extend this action to timbre. CA have also been used to model timbre in various ways. For example, in the work of Miranda ([10], [11]) 2-dimensional CA and granular synthesis are used to generate timbres which follow some of the natural properties of acoustically generated sounds. Miranda says: "the cellular automaton models the way in which most natural sounds produced by acoustic instruments evolve: they tend to converge from a wide distribution of their partials to form oscillatory patterns." The grains which are used as "building blocks" are typically $30-40$ ms in length. In our work we also use small segments, or "cycles", about $1-20$ ms of digital audio, represented with a polynomial spline model. The timbre evolves through a sequence of cycles which can be generated with CA, and which can also progress from chaotic spectrum (attack phase) to narrower band oscillatory patterns. We focus here on 1-dimensional CA.

We begin our description of CA as a *local rule* which generates a sequence of n-dimensional vectors

$$\mathbf{v}_k = (c_0, c_1, \ldots, c_{n-1}).$$

This means that we generate vectors iteratively according a function F so that

$$\mathbf{v}_{k+1} = F(\mathbf{v}_k)$$

in some neighborhood $\mathcal{N}_i$ of each coordinate c_i, for $i = 0, \ldots, n-1$. The neighborhood $\mathcal{N}$ specifies a set of indices k containing i. For our application to audio cycles, we suppose that each such vector consists of the B-spline coefficients for one cycle. In order to render audio samples from one such cycle, we evaluate the linear combination of B-splines, the spline function

$$f(t) = c_0\mathscr{B}_0(t) + \cdots + c_{n-1}\mathscr{B}_{n-1}(t)$$

at each of the sample input values t. A typical case could be, for instance, a cycle consisting of 100 output samples generated from a function $f(t)$ with $n = 20$ cubic B-spline coefficients, at sample rate 48 kHz, producing a tone with fundamental frequency 480 Hz.

A simple type of local rule used to define a CA is to average neighboring values. For example, if we denote the coordinates of $\mathbf{x}_{k+1}$ as c'_i:

$$c'_i = \frac{1}{2}(c_i + c_{i-1}).$$

This local rule also can be interpreted as a filter. (To evaluate for $i = 0$ we assume $c_{-1} = 0$.) If we apply this local rule directly to samples, we have a familiar low-pass filter (transfer function $\mathscr{H}(z) = \frac{1}{2}(1+z^{-1})$), but if we apply it to the B-spline coefficients of one cycle the result is slightly different. The result can be seen through the process of B-spline evaluation with the de Boor algorithm. With degree $d = 3$, each output sample is the result of nested linear interpolation starting from 4 B-spline coefficients. A delay by one B-spline coefficient then has the effect of a delay by one subinterval on the time axis, but only within one cycle. If the subintervals are evenly spaced, this will in turn have the effect of a delay by α samples within the cycle, where α is the number of samples per subinterval, which is not necessarily an integer. If α is in fact an integer, then this filter is similar to an inverse comb filter, with magnitude frequency response having notches at $\frac{f_N}{\alpha}(2i+1)$ for integers i and Nyquist frequency f_N. However, since the local rule is applied one cycle at a time, it does not perform exactly as a filter which is typically applied continuously to an audio stream.

Next we consider another familiar class of CA called Elementary Cellular Automata, which are defined and illustrated here [12]. These CA are set up to act on vectors of 0's and 1's, which can be easily adapted to apply to B-spline coefficients. They are locally defined using one cell and its two neighbors. The output is 0 or 1 depending on the binary digits of a positive integer r, with $0 \le r \le 255$. Let r be written as an 8-digit binary expansion: $r_7r_6\ldots r_0$ (so that r_7 is the binary coefficient of 2^7) and let the current vector $\mathbf{x}$ be represented as a binary string $x_0x_1\ldots x_{n-1}$. In order to process consecutive triples, also assume we have $x_{-1} = x_n = 0$. Then for each cell (or coordinate) x_i we consider the triple $x_{i-1}x_ix_{i+1}$ as the binary expansion of an integer $b(i)$ from 0 to 7. The local rule then assigns the value $r_{b(i)}$ as the output x'_i. Denote this Elementary CA as $ECA(r)$. We illustrate the local rule for $ECA(30)$ in Table 1. The first row gives the 8 possible triples $x_{i-1}x_ix_{i+1}$ and the second row is the output $r_{b(i)}$.

111	110	101	100	011	010	001	000
0	0	0	1	1	1	1	0

Table 1. local rule for ECA(30)

```
                    1
                   111
                  11  1
                 11 1111
                11  1   1
               11 1111 111
              11  1    1  1
             11 1111  111111
            11  1   111     1
           11 1111 11  1   111
          11  1    1 1111 11  1
         11 1111  11 1    1 1111
        11  1   111  11  11 1   1
       11 1111 11  111 111  11 111
      11  1    1 111   1  111  1  1
     11 1111  11 1  1 111111  1111111
    11  1   111  1111 1     111      1
   11 1111 11  111    11  11  1    111
  11  1    1 111  1  11 111 1111  11  1
 11 1111  11 1  111111  1   1   111 1111
11  1   111  1111     1111 111 11   1   1
```

Table 2. output from ECA(30), $n = 41$

As in [12], we take as initial input sequence to $ECA(r)$, a vector $\mathbf{x}$ of length n odd, with $x_i = 1$ for $i = \frac{n-1}{2}$, and $x_i = 0$ otherwise. In Table 2, we use $n = 41$ and replace 0's with spaces for ease of visualization.

We apply $ECA(r)$ to B-spline coefficients c_i by multiplication with $a + bx_i$ for some choice of values a and b with $a + b = 1$. This allows the generation of key cycles which are derived from a particular fixed cycle C_0, with B-spline coefficients c_i, $i = 0, \ldots, n-1$. If $a = 1$ then there is no change to the cycles. If $b = 1$ then we have the maximal change as the B-spline coefficients are turned on or off according to the value of x_i in the sequence. Another approach is to borrow a sequence of key cycles from a recorded sound and to apply $ECA(r)$ iterates to this sequence. Again, the amount of change to the cycles can be modified by the choice of a and b. In this way, timbre is modified through the action of CA on the B-spline coefficients of cycles. In the next section we see how this notion of effecting change in other musical elements can also be described through the action of various CA.

5. MULTIDIMENSIONAL MUSIC SEQUENCE

In order to describe the action of CA on music sequences, we begin with the general definition of CA. In general, a 1-D **deterministic cellular automaton** is described by a neighbourhood $\mathcal{N} \subset \mathbb{Z}$, a set $\mathbb{A}$ called an alphabet, with $\mathcal{X} = \mathbb{A}^{\mathbb{Z}}$, and a transformation $F : \mathcal{X} \to \mathcal{X}$ defined by its **local function** $f : \mathbb{A}^{\mathcal{N}} \to \mathbb{A}$ by the relation :

$$F(\boldsymbol{x})_i = f\big((x_{i+k})_{k\in\mathcal{N}}\big) = f\big(x_{i+k_1}, \ldots, x_{i+k_n}\big).$$

For 2-D automata, $\mathbb{Z}$ must be replaced by $\mathbb{Z}^2$, and for 3-D automata by $\mathbb{Z}^3$. A simple but essential CA is called the **shift map** acting the following way : $\forall \boldsymbol{x} \in \mathbb{A}^{\mathbb{Z}} : \sigma(\boldsymbol{x})_i = x_{i+1}$. For every CA F we have that $F \circ \sigma = \sigma \circ F$. We introduce two automata with $\mathcal{N} = \{0,1\}$, $\tau = \sigma + \mathrm{id}$ and $\Delta = \sigma - \mathrm{id}$. The first one is called the *Ledrappier automaton* and the second one the *Vieru automaton*. Those two CA have been widely studied in [13, 14]. $f_\tau(a,b) = a + b$ and $f_\Delta(a,b) = b - a$.

Next, we define a music sequence as a sequence of sound events, each of which can be described by pitch α, duration β, and timbre γ. We restrict here to these parameters simply to illustrate with a basic example. Each of these elements can be described in an absolute sense, or a relative sense. For pitch, for instance, the absolute sense could be given as fundamental frequency in Hz, or as a letter name such as $A4$ to indicate fundamental frequency restricted to the equal temperament system based on $A4 = 440$ Hz. The relative sense for pitch describes the interval compared to the previous pitch, which could be given by a frequency ratio such as $\frac{3}{2}$, or a musical interval interval name such as Just Perfect Fifth. In the equal temperament system intervals can be simply indicated as an integer number of semitones, or if the sequence of pitches is considered under octave equivalence, as integers modulo 12, or $\mathbb{Z}_{12}$. In a similar way we can express the time values of notes in absolute or relative time (or duration) since the previoius note onset. All of these concepts are of course displayed in traditional music notation. The element of timbre, however, tends to be treated quite differently. In the broadest sense, when one scores for orchestra there is a discrete set of instruments representing different timbres. Also, solo instruments, such as bowed or plucked strings, have timbre markings such as *sul ponticello* or *sul tasto*, indicating notes to be played near the bridge or over the fingerboard with the resulting change in timbre. All of these notions of timbre are typically thought of as absolute, meaning that a note is either played by a flute or violin, or perhaps a note is indicated as *sul ponticello* on a cello. Relative changes in timbre can often be left up to the performer. With the approach of CA applied to our spline models, it is possible to introduce relative changes to timbre both within individual notes and for a sequence of notes.

Next we present an example of a music sequence based on relative measure of pitch, rhythm and timbre. For simplicity, we assume a discrete time axis with basic unit of time given as one eighth note, and we use a binary string to indicate the onset of a note as 1 and the absence of a note, or silence, as 0. Notes are assumed to be in equal temperament, and intervals between notes are referred to by number of semitones. To make the rhythmic pattern a bit more interesting, we introduce the famous Fibonacci word,

$$\boldsymbol{t}_{\text{fib}} = 0100101001001010010100100101001001\ldots$$

which can be defined recursively by concatenation

$$S_i = S_{i-1}S_{i-2}$$

with $S_0 = 0$ and $S_1 = 01$. This Fibonacci word is a Sturmian sequence, which means that it is aperiodic and balanced (having precisely $n+1$ distinct factors, or subwords, of length n for any positive integer n.) Also, let $\mathbf{1}$ be the constant sequence of 1's, so that $\boldsymbol{t}_{\text{fib}} + \mathbf{1}$, with addition mod 2, is the complement to $\boldsymbol{t}_{\text{fib}}$.

Now we define two music sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ by first specifying pitch and rhythm. Suppose that the pitch of $\boldsymbol{x}$ simply alternates between C and E four semitones higher. As a relative pitch sequence we would then have

$$\alpha_{\boldsymbol{x}} = \{4, -4, 4, -4, \ldots\}.$$

Suppose also that the onsets of the notes for $\boldsymbol{x}$ are given by the 1's in the sequence $\boldsymbol{t}_{\text{fib}} + \mathbf{1}$. Then the relative sequence of note durations for $\boldsymbol{x}$, in numbers of eighth notes, is

$$\beta_{\boldsymbol{x}} = \{2, 1, 2, 2, 1, 2, \ldots\}.$$

Now suppose that the pitch of $\boldsymbol{y}$ simply increases by major thirds, starting with $B\flat$, so that the relative pitch sequence is

$$\alpha_{\boldsymbol{y}} = \{4, 4, 4, \ldots\},$$

and that the onsets for the notes in $\boldsymbol{y}$ are given by the sequence $\boldsymbol{t}_{\text{fib}}$. Then the relative sequence of note durations for $\boldsymbol{y}$ is

$$\beta_{\boldsymbol{y}} = \{3, 2, 3, 3, 2, 3, \ldots\}.$$

The sequences $\mathbf{x}$ and $\mathbf{y}$ are realized on the staff in figure 3, allowing for octave equivalence. In this example we can think of $\boldsymbol{x}$ and $\boldsymbol{y}$ as two initial subjects which are then modified using Δ and τ to produce (very simple) countersubjects which enter one measure later in upper part of the staff. Also note: the starting pitches in each voice are somewhat arbitrary, the focus being on relative pitch in the sequences. (See figure 3.)

Next, we need to describe the timbre for $\boldsymbol{x}$ and $\boldsymbol{y}$. Suppose we choose an initial wave form, based on a single cycle C_0, such as the one at the top in figure 4. Such a cycle can be used with a simple envelope to specify one note at a given fundamental frequency or pitch. The simplest case would be to use this one cycle for all the notes in the example, so that the relative change in timbre is zero, or no change. This would mean the timbre sequence

$$\gamma_{\boldsymbol{x}} = \gamma_{\boldsymbol{y}} = \{0, 0, 0, \ldots\}.$$

Can we change the timbre by indicating a single value to replace 0? Typically timbre is much more complicated, but to continue this basic example, we could simply adjust one B-spline coefficient. For example, suppose we adjust c_r for a fixed choice of r, with $0 < r < n-1$, by some small non-zero value. If we continue to do this, making small changes to c_r in the cycle C_0 as we progress through the notes, we will have a sequence of notes with slightly different timbres and also a simple numeric sequence giving the relative changes in timbre. For example, let's suppose that

$$\gamma_{\boldsymbol{x}} = \{.01, .01, .01, \ldots\}$$

and

$$\gamma_{\boldsymbol{y}} = \{-.01, -.01, -.01, \ldots\}.$$

Finally, consider the action of Δ and τ on $\boldsymbol{x}$ and $\boldsymbol{y}$. Each action is applied to α, β and γ, by the local rules:

$$\Delta(\boldsymbol{x})_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i \text{ and } \tau(\boldsymbol{x})_i = \boldsymbol{x}_i + \boldsymbol{x}_{i+1}.$$

When CA are used on various musical elements, the action splits.

$$\begin{tikzcd}
 & \Delta(\mathbf{x}) = \Delta(\alpha_{\mathbf{x}}, \beta_{\mathbf{x}}, \gamma_{\mathbf{x}}) \arrow[dl] \arrow[d] \arrow[dr] & \\
\Delta(\alpha_{\mathbf{x}}) & \Delta(\beta_{\mathbf{x}}) & \Delta(\gamma_{\mathbf{x}})
\end{tikzcd}$$

Figure 2. CA action splitting

In the case of pitch, we see that

$$\Delta(\alpha_{\boldsymbol{x}}) = \{-8, 8, -8, 8, \ldots\}, \tau(\alpha_{\boldsymbol{x}}) = \{0, 0, 0, \ldots\},$$

$$\Delta(\alpha_{\boldsymbol{y}}) = \{0, 0, 0, \ldots\}, \text{ and } \tau(\alpha_{\boldsymbol{y}}) = \{8, 8, 8, \ldots\}.$$

In the case of rhythm, we apply the rule directly to the binary sequence and use addition mod 2, which gives the same result for all four:

$$\Delta(\beta_{\boldsymbol{x}}) = \Delta(\beta_{\boldsymbol{y}}) = \tau(\beta_{\boldsymbol{x}}) = \tau(\beta_{\boldsymbol{y}})$$

$$= \{1, 1, 0, 1, 1, 1, 1, 0, \ldots\}.$$

In the case of timbre, we have:

$$\Delta(\gamma_{\boldsymbol{x}}) = \{0, 0, 0, \ldots\} = \Delta(\gamma_{\boldsymbol{y}}),$$

$$\tau(\gamma_{\boldsymbol{x}}) = \{.02, .02, .02, \ldots\} \text{ and}$$

$$\tau(\gamma_{\boldsymbol{y}}) = \{-.02, -.02, -.02, \ldots\}.$$

The above is illustrated in the following very simple multidimensional musical fragment in figure 3.

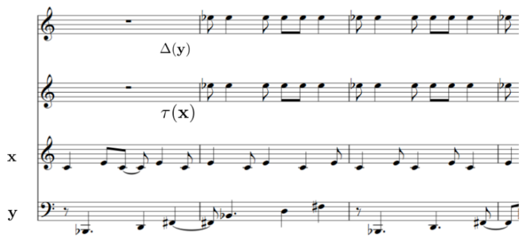


Figure 3. Multidimensional example

6. PROBABILISTIC AND OTHER TYPES OF CA

We saw in the application of $ECA(r)$ to B-spline coefficients that it can be useful to choose constants a and b to form a multiplier other than simply 0 or 1 to produce changes. This approach can be taken a step further by introducing random variables into the CA computations, which are called **probabilistic cellular automata**. We introduce here the probabilistic versions of Δ and τ called Δ_ε and τ_ε. The idea is simply to perform the local rule update with probability $1-\varepsilon$. For example, the process of updating the n B-spline coefficients of one cycle now involves a sequence of flips of a biased coin, with probability of Heads being $1-\varepsilon$. For each coefficient, the output will be determined by the (deterministic) CA if the flip is Heads, and otherwise the coefficient is not updated. Hence the number of B-spline coefficients which experience a change in this process is a binomial random variable.

We can also add small perturbations p to the B-spline coefficients to obtain further probabilistic CA's. For example, define $\Delta_{\varepsilon,p}$ to have the same function as Δ_ε, except that when the local rule is applied we are also adding a small value, or perturbation, p to the B-spline coefficient. We can set p to be constant for a sequence of cycles generated from this CA, or we could also replace p by X_p, a random variable with uniform distribtion on the interval $[-p,p]$. Thus the CA Δ_{ε,X_p} would have local rule applied to any B-spline coefficient with probability $1-\varepsilon$ give as:

$$c_i' = c_i - c_{i-1} + X_p$$

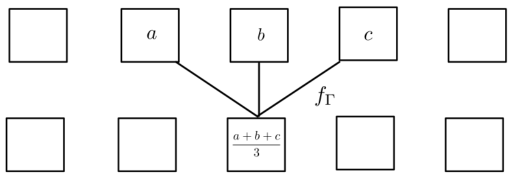
where $X_p \in [-p,p]$.

In figure 4 we show the comparison of Δ and Δ_ε applied to a cycle taken from a guitar pluck recording. Here we use the value $\varepsilon = 0.2$.

We can also define the automata Γ, similar to τ by

$$\Gamma(\boldsymbol{x})_i = f_\Gamma(x_{i-1}, x_i, x_{i+1}) := \frac{x_{i-1}+x_i+x_{i+1}}{3}.$$

We describe its action on cells :



Again, Γ behaves as an inverse comb filter on the signal values generated by the spline function, on the portion of one cycle which has uniform knot sequence. We illustrate the differences between Γ and Γ_{ε,X_p} on the same initial cycle in figure 5, with values $\varepsilon = 0.2$ and $p = 0.3$.

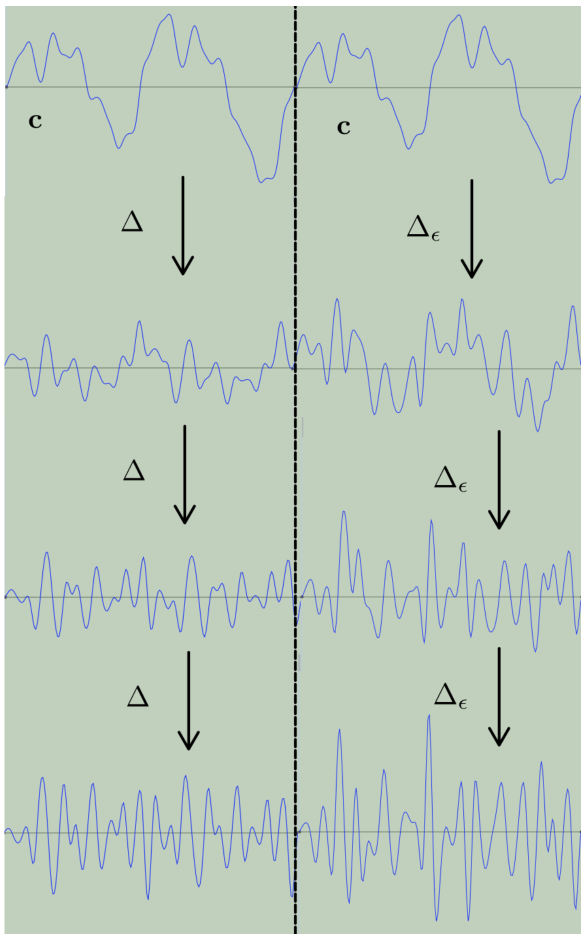


Figure 4. One cycle evolving through Δ and Δ_ε

6..1 Cycle Interpolation (CI) Types

A nice model of synthesized sound can be developed from these ideas. We begin with one cycle C_0, perhaps taken from a recorded sound or generated with sinusoids, or a particular selection of initial B-spline coefficients. Such a cycle can be thought of on the level of one grain, in comparison to granular synthesis. Next, specify an envelope, which can also be borrowed from an actual recorded sound, or can be formed as ADSR or other methods. The waveform is then generated from these intial data by the use of CA and CI. This assumes a choice of the placement of key cycles (regular or otherwise) with all intermediate cycles to be generated by CI. Finally, the CI process also involves the choice of meta-splines, cubic or linear for instance. If the CA is τ, then the iteration of cycles $C_j = \tau(C_{j-1})$. Thus, the gradual evolution of timbre is formed by both CA and CI. We will refer to the sequence of cycles generated by the above method as *Type I*. As a special case, one could also apply this method to every cycle of a sequence, with no CI. In figure 6 cycles 0, 5 and 10 are key cycles, and the rest use CI with piecewise linear meta-splines. Cycle 0 is taken from an audio sample of a guitar pluck at roughly A440, and cycles 5 and 10 are iterates of this cycle using the automata Γ_ε defined above.

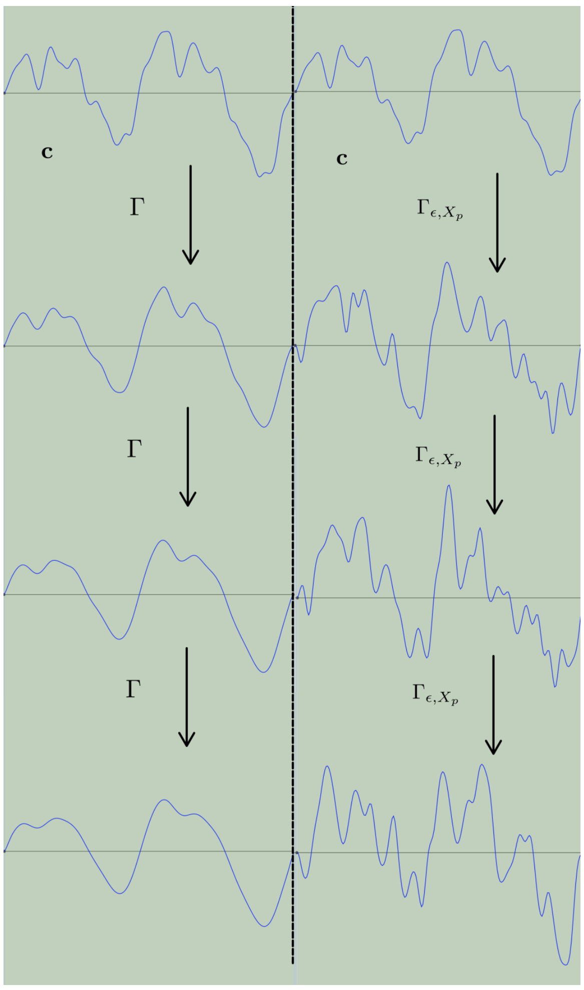


Figure 5. One cycle evolving through Γ and Γ_{ε,X_p}

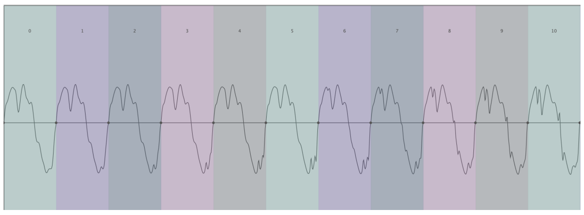


Figure 6. Type I CA with CI example

6..2 Higher scale action

In the above we have applied the CA as a local rule acting on one cycle's B-spline coefficients. We can also instead apply this local rule across cycles but focusing on one B-spline coefficient. For instance, the rule to change the i^{th} B-spline coefficient c_i^j of cycle C_j could use the triple of coefficients

$$c_i^{j-1},\ c_i^j,\ c_i^{j+1}.$$

Doing this for all $i = 0,\dots,n-1$, and $j = 1,\dots,n-2$ we can modify a sequence of cycles. This can be applied either to the key cycles, or to an entire sequence of cycles.

This can be interesting to apply to an initial sequence which was not of Type I. This allows the set of key cycles K now to be generated by any process, whether they are taken from various audio samples, or specific basis choices, or random. Call this intial sequence K_0. Then we can iterate the CA to form new sets of key cycles K_r, and in turn generate new sequences of cycles by CI. A sound sample generated in this way will be called *Type II*. As with Type I, we also have the special case where there is no CI, or equivalently every cycle is key.

7. CONCLUSIONS

We saw that this work takes its origin from the concrete context of UPISketch. The CA were already used on pitches and rhythms to generate new material, starting from an initial configuration. We presented here an approach to generate audio signals with evolution of cycles. The method of combining cellular automata and cycle interpolation is capable of modeling timbres which are close to realistic sounds, such as musical instruments, and also designing new and interesting timbres. We have an implementation, using JUCE, to illustrate both the graphical and audio examples discussed in this paper. We are also motivated to contribute convincing audio examples in the UPISketch software, showing the design techniques presented in this paper. Further, this work continues the goal of exhibiting the multi-dimensional action of the CA on musical elements.

8. References

[1] Bourotte, R., Kanach, S., *UPISketch: The UPIC idea and its current applications for initiating new audiences to music*. Organised Sound, vol 24, no.3, 252-260, 2019.

[2] Hajda, J., M., *The Effect of Dynamic Acoustical Features on Musical Timbre*. Analysis, Synthesis, and Perception of Musical Sounds, J.W. Beauchamp, editor, Springer, 2013, ISBN 10: 0-387-32496-8, pp 250-271.

[3] Donnadieu, S., *Mental Representation of the Timbre of Complex Sounds*. Analysis, Synthesis, and Perception of Musical Sounds, J.W. Beauchamp, editor, Springer, 2013, ISBN 10: 0-387-32496-8, pp 272-319.

[4] Steiglitz, K.: *A Digital Signal Processing Primer, with Applications to Digital Audio and Computer Music*, Addison-Wesley, (1996).

[5] Collins, N.: *SplineSynth: An Interface to Low-Level Digital Audio*. Proceedings of the Diderot Forum on Mathematics and Music, Vienna, 1999, ISBN 3-85403-133-5, pp 49-61.

[6] Ardaillon, L., Degottex, G., Roebel, A.: *A multi-layer F0 model for singing voice synthesis using a B-spline representation with intuitive controls*. Interspeech 2015, Sep 2015, Dresden, Germany. hal-01251898v2.

[7] Klassen, M.: *Spline Modeling of Audio Signals and Cycle Interpolation*, Mathematics and Computation in Music, MCM 2022, `https://azrael.digipen.edu/research/`

[8] de Boor, C.: *A Practical Guide to Splines*, revised edition, Springer-Verlag, New York (1980)

[9] Lanthier, P.: *Multi dimensionnal action of cellular automata on music sequences*, preprint, 2022. `https://azrael.digipen.edu/research/`

[10] Miranda, E.R. (2003) *Evolving Cellular Automata Music: From Sound Synthesis to Composition*, `https://www.researchgate.net/publication/228862474`

[11] Miranda, Eduardo R., and Alexis Kirke. ”Game of life music.” Game of Life Cellular Automata. Springer, London, 2010. 489-501.

[12] Weisstein, Eric W. ”Elementary Cellular Automaton.” From MathWorld–A Wolfram Web Resource. `https://mathworld.wolfram.com/ElementaryCellularAutomaton.html`

[13] Lanthier, Paul and Guichaoua, Corentin and Andreatta, Moreno (2019) *Reinterpreting and Extending Anatol Vieru’s Periodic Sequences Through the Cellular Automata Formalisms*, Mathematics and Computation in Music, Springer International Publishing.

[14] Lanthier, Paul (2020), *Aspects ergodiques et algébriques des automates cellulaires*, Université de Rouen, PhD thesis.