Synthesis Idea – Continuous Deformation

Contents

1 Concept.

Consider a single cycle of a clickless periodic waveform

$$\omega : [0, 2\pi] \to [-1, 1]$$

satisfying $\omega(0) = \omega(2\pi) = 0$.

I would like a way to generate a sequence of functions

$$\begin{gathered} f_1 : [-1,1] \to [-1,1] \\ f_2 : [-1,1] \to [-1,1] \\ \vdots \\ f_N : [-1,1] \to [-1,1] \end{gathered}$$

such that for all n:

1. The result,
$$f_n \circ \omega : [0, 2\pi] \to [-1, 1],$$
is also a single cycle of a clickless periodic waveform with endpoints equal to zero.

2. The output sounds smooth when changing from f_n to f_{n+1}.

Assigning n to a parameter knob, we could continuously deform the waveform as follows:

$$\omega \leftrightarrow f_1(\omega) \leftrightarrow f_2(\omega) \leftrightarrow \cdots \leftrightarrow f_N(\omega).$$

Hopefully, sometimes, this would sound good. By experimenting with different sequences (f_n), my hope is that I could create a number of novel ways to continuously transform a waveform that would not be easy or possible through conventional synthesis methods.

2 Discrete approach.

This is all hugely simplified by the Nyquist–Shannon sampling theorem, which lets us work with ω and the f_n as discrete functions. Let's use 24-bit audio, and let A be the corresponding sample range

$$A = \{-1, -1+2^{-23}, -1+2\cdot 2^{-23}, -1+3\cdot 2^{-23}, \ldots, 1-2^{-23}, 1\}.$$

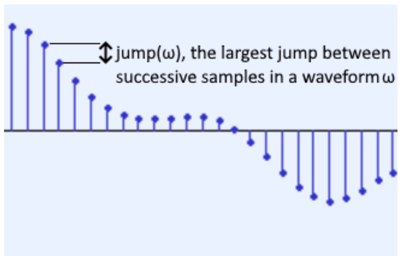
This gives the following:

$$\omega : \{1, 2, \ldots, 44100\} \to A$$
$$f_n : A \to A$$

I think the following definitions make things a bit easier.

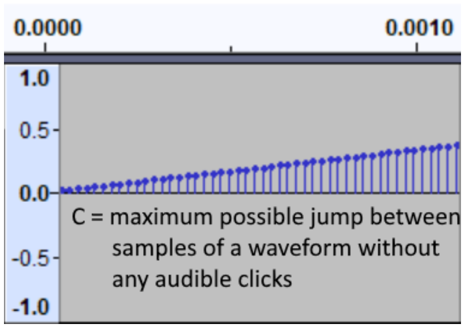
Definition 2.1. Define $\operatorname{jump}(\omega)$ as

$$\max_n |\omega(n+1) - \omega(n)|.$$


jump(ω), the largest jump between successive samples in a waveform ω

Define

$$C = \max_{\text{all } \omega} \operatorname{jump}(\omega).$$



I think we'll have to test this empirically ourselves to find our value for C, but maybe there's some scientific way to determine its "true" value.

2.1 Conditions

I can now rewrite the two conditions above for this discrete form. For all $f_n : A \to A$, we need the following.

1. $\text{jump}(f_n \circ \omega) < C$ and $f_n \circ \omega(0) = f_n \circ \omega(44100) = f_n(0) = 0$.

   This means f_n must satisfy:

   - For all $a, b \in A$,
     $$|a - b| < C \Rightarrow |f_n(a) - f_n(b)| < C.$$
     This means, if clickless audio goes in, clickless audio has to come out.
   - $f_n(0) = 0$ always.

2. The output sounds smooth when changing from f_n to f_{n+1}.

   Two approaches occurred to me here.

   (a) Set a global limit ϵ on how much f_{n+1} can differ from f_n, and demand that for all n,
   $$\max_x |f_{n+1}(\omega(x)) - f_n(\omega(x))| < \epsilon,$$
   where ϵ is small enough that making the change sounds smooth.

   (b) More complex, but possibly interesting: f_1 can depend on ω, and similarly f_n can depend on $f_{n-1}(\omega)$ for $n > 1$.

   The idea with this is that, instead of setting a global limit, you set lots of local limits based on the waveform you are transforming. This way, the limits can be chosen to be as unrestrictive as possible while still making the change between successive functions sound smooth. See §A.2 for more on this.

3 Implementation

There are a few different directions which seem promising to me. I think that experimentation rather than theory is the way to go here as we may discover things that are technically not allowed but sound good anyway, despite or because of that.

As with Synplant, rather than controlling the process directly, I would like to be able to algorithmically generate a ton of functions, choose what works, and discard the rest. This principle informs the approaches outlined below.

3.1 f_n independent of ω

1. Choose values for C and ϵ.
2. Generate a list of candidate functions $s_{1,i}$ for f_1, such that for all $s_{1,i}$ we have:
   - For all $a, b \in A$, $|s_{1,i}(a) - s_{1,i}(b)| < C$.
   - $s_{1,i}(0) = 0$.
   - For all n,
     $$\max_x |s_{1,i}(\omega(x)) - \omega(x)| < \epsilon.$$
3. Similarly, for $1 < k \leq N$, generate a list of candidate functions $s_{k,i}$ for f_k, such that for all $s_{k,i}$ we have:
   - For all $a, b \in A$, $|s_{k,i}(a) - s_{k,i}(b)| < C$.
   - $s_{k,i}(0) = 0$.
   - For all n,
     $$\max_x |s_{k,i}(\omega(x)) - s_{k-1,i}(\omega(x))| < \epsilon.$$
4. We now have a selection of different parametrisable sequences
   $$(s_{n,i}) = (s_{1,i}, s_{2,i}, \dots, s_{N,i}).$$
   Try them out and see if any sound good. For example, the first sequence $(i = 1)$ is
   $$\omega \leftrightarrow s_{1,1}(\omega) \leftrightarrow s_{2,1}(\omega) \leftrightarrow \cdots \leftrightarrow s_{N,1}(\omega).$$
   As discussed in §A.1, you can plug different sequences into each other if you so wish.
5. Save any good sequences, throw away the rest, and repeat.

I haven't bothered writing out how to generate the candidate functions yet, but was just thinking of using a probabilistic method. Could start out simply with `random()` (uniform distribution), and then using other methods or probability distributions if we suspected they might be more computationally efficient or yield a higher proportion of good results.

3.2 Manual method

Outline: choose a sequence of different nice-sounding functions $f_a, f_b, \dots, f_z$ then interpolate between them to create a smooth sequence

$$\omega, f_1(\omega), f_2(\omega), \dots, f_k(\omega), f_a(\omega), \dots, f_b(\omega), \dots, f_z(\omega).$$

Algorithm:

1. Generate a list of candidate functions s_i for f_1, such that for all s_i we have:
   - For all $a, b \in A$, $|s_i(a) - s_i(b)| < C$.
   - $s_i(0) = 0$.
2. Choose a function to be used for f_a, such that $f_a(\omega)$ sounds good.
3. Choose a value for k and ϵ, then interpolate with your preferred algorithm to generate a sequence

   $$\omega, f_1(\omega), f_2(\omega), \ldots, f_k(\omega), f_a(\omega)$$

   such that, for all n,

   $$|f_{n+1}(\omega(x)) - f_n(\omega(x))| < \epsilon,$$

   i.e. that you can change smoothly from ω through to $f_a(\omega)$ using this sequence.
4. Repeat steps 1–3 until you have a sequence of desired length.

A Appendix

A.1 Composing multiple sequences

Let Ω be the set of all clickless periodic waveforms with endpoints equal to zero, so that $\omega \in \Omega$. Then for all f_n, we also have $f_n \circ \omega \in \Omega$.

This means if we have multiple distinct sequences of functions, say (f_p), (g_q), (h_r), that satisfy conditions §2.1, we can compose them as follows:

$$h_r \circ g_q \circ f_p \circ \omega$$

and use them as a function F with multiple parameters that could be assigned to knobs:

$$F(p, q, r) = h_r((g_q(f_p(\omega))))$$

The only limits to the number of functions that can be composed at one time are processing power and memory.

A.2 Local limits

I had vague idea of how to make the system for choosing ϵ more configurable but have no idea if it would be good or not so put it here.

The idea is that we place different limits on different functions, depending on the previous waveform. The aim is still to make changing between functions sound smooth, but the idea is that we are as unrestrictive as possible. So for example, we might find that

$$|f_1(\omega(x)) - \omega(x)| < \epsilon$$

is the biggest limit we can impose for the change between ω and $f_1(\omega)$ to sound smooth, but that the less restrictive limit

$$|f_{10}(\omega(x)) - f_9(\omega(x))| < 2\epsilon$$

works for the change between f_9 and f_{10}.

We might also find that we can have multiple limits within a single function. Depending on the input waveform, we might find a limit of ϵ is all we can use from $[-1, 0]$, but that a limit of 2ϵ works for $[0, 1]$. This approach is more complex than §3.1 and I don't want to try it yet, but have placed the section here to remind myself that it could potentially be interesting.