Topology-Preserving Deformations of Streaming Audio via Box Snake Surgery

Georg Essl
University of Wisconsin - Milwaukee
essl@uwm.edu

Robin Belton
Vassar College
rbelton@vassar.edu

ABSTRACT

Topology provides both strong and principled invariants as well as defines spaces of deformability. We extend a method for segmentation and deformation of fixed length audio sample data to the case of streaming data. This method uses sublevel set persistent homology to segment audio data into extremal and monotone blocks called box snakes, *where monotones can be deformed so long as monotonicity is preserved. We employ surgery on box snakes to realize shifts of data. We discuss this process with respect to both linear and circular domains. This allows box snake structures to be matched with the topologies of other audio signal processing techniques such as the Discrete Fourier Transform which naturally operates on a circular topology.*

1. INTRODUCTION

Sound synthesis and audio manipulation methods operate on a space of variability of sound under certain chosen properties. In recent years, topological properties have been proposed as useful for providing both variability and specific rigidity to sound manipulation.

One such topological method for audio manipulation was recently proposed using sublevel set persistent homology [1]. Extrema (minima, maxima) in audio signals are points of topological change with regard to the "height" or level of the audio data. It turns out that monotone sequences do not change the topology, and hence can be deformed as long as they stay monotone. Previous work [1] was limited to a fixed length sequence of audio samples, and hence was applicable to looped audio with a static data buffer. The purpose of the work described here is to show how this limitation can be removed and hence make the method amenable to streaming audio data.

This paper belongs to a growing body of work that explores topological methods for sound synthesis, either by giving more flexibility to existing synthesis methods [2] or by providing direct deformation strategies of oscillators [3]. Topology describes properties that on the one hand are global, and therefore very robust, and on the other hand local, and on that level flexible. Audio signals can be constructed [1] that allows principled deformation rules that respect global properties while exploiting, in a well-defined way, local deformability. One can fruitfully think of topological properties as describing a kind of invariant of the signal under deformations. The approach proposed in [1] is attractive because it operates directly on audio signals, which distinguishes it from previous methods which utilize geometric settings that are higher dimensional, such as oscillators embedded in a plane [3], or winding paths on the surface of a torus in three dimensions [2].

The central data structure of [1] consists of a block-wise subdivision called *box snakes*. Box snakes segment the audio data in such a way as to identify extrema, as well as segment monotone sequences, which describes the space of deformability of audio signals. This can be viewed as a topology-preserving technique, with the topology under consideration being described by sublevel set persistent homology [4]. In this paper, we show that a set of surgery procedures on box snakes allows one to implement a streaming realization of the algorithm. Furthermore we explore streaming linear and circular sliding buffers and their respective properties. It is well known that the Discrete Fourier Transform (DFT) involves circular convolutions [5, 6], which can be explained topologically via Pontryagin duality [7, 8].

Circular buffers for sublevel set persistent homology has nice symmetry properties, which leads to natural segmentation of periodic functions within a window and avoids artifacts due to boundary effects. This makes it an attractive choice for streaming and windowed processing. Any circular domain can be converted into a linear domain with a single cut surgery. This relationship allows the combination of linear and circular domains in processing pipelines involving box snakes [1].

2. RELATED WORK

This work is most directly related to work on dynamically updating persistence information [10], which assumes that extrema are isolated. This means that no two extrema share the same level, and that extrema only occupy one sample and do not form flat regions. This restricted class of functions are known as Morse functions. To ensure isolated extrema, samples are either perturbated, or tie-breaking rules are employed to disambiguate extrema at the same level. However, audio data is not guaranteed to have non-identical extrema in its data. In fact, a constant function, representing silence, as well as any sinusoidal function has, by definition, multiple extrema at the same value. Furthermore, flat extrema can occur in audio data. Square waves are a common synthetic example. Hence it is desirable to require none of the typical restrictions of Morse functions. These restrictions can be overcome [1], and this is

[1] Our discussion will be self-contained. A detailed discussion of the underlying theory can be found in [9].

discussed in detail in [9]. Another common assumption in prior work concerns bar code construction rules. The dominant rule, as used in [10] is well known as the *elder rule*. It says that when two (or more) connected components merge at a maximum, the first created component continues while the other "dies". This corresponds to the lowest minimum associated with the connected components. This rule is, however, arbitrary and can be replaced by other rules. As the lowest minimum and the global maximum can be arbitrarily placed in a sequence of audio, the elder rule leads to barcode updates that do not match the streaming pattern, and, in the worst case, can lead to bars that span the whole length of a sequence. This may cause the bars to re-arrange every streaming update, and thus lead to erratic barcode patterns that make it hard to interpret data.

To address this problem, we propose the use of circular domains together with a local barcode construction rule to ensure a notion of temporal coherence of a barcode. Our way to update an auxiliary data structure also differs, in that we use the box snake structure [1, 9] rather than the multi-pane windows [2] structure proposed in [12] and dynamically updated in [10].

This work contributes to the increasing body of work in topological methods in signal processing [13, 14] and its applications to sound synthesis [2, 3], digital signal processing [15, 16, 17], and the design of musical interfaces [18, 19]. Numerous applicable techniques have been proposed for general time series data such as topological analysis based on embedding data in higher dimensional spaces [20]. This has been used for differentiating a clarinet from a viola tone [21], and detecting wheeze sounds [22].

In this paper, we contribute a streaming extension of processing of audio data based on snake boxes, surgery of snake boxes, and deformation on streaming audio. A particular contribution of this work is the interconnection of sublevel set persistence of audio data with discrete Fourier transform processing (DFT/FFT). This extends the static looped case frequency-domain manipulations previously considered [1]. Finally, we discuss examples of streaming single-stage as well as multi-stage audio processing pipelines.

3. LINEAR AND CIRCULAR DOMAINS

Conventionally, audio data is thought of as linearly ordered with respect to time. Yet perception of audio appears to parse repetition [23], and especially in the context of fast convolutions and other techniques relying on the Fast Fourier Transform (FFT) as perhaps the most important variant of the Discrete Fourier Transform (DFT) the circular (also sometimes called cyclically ordered) domains also need to be considered, and the relationship of linear processing of short-time Fourier transforms (STFT) and FFTs became part of the construction. The relationships between an infinitely extended linear domain and finite circular domain of DFTs is depicted in Figure 1. The figure depicts the time-domain on the left, and the frequency-domain under the appropriate Fourier transform on the right. Topologically, the duality properties between transformed domains are known as Pontryagin duality [7, 8]. The key theorem of Pontryagin duality for this picture states that if the domain (characterized as a topological group) is compact, then the dual necessarily must be discrete and vice versa. We will call this theorem the *compact-discrete theorem*. Compactness has a technical definition, but for our purpose, we will work with the intuitive notion that if a signal domain extends to infinity, then it is not compact. Otherwise, the signal domain is compact. For example, the real line and the full set of integers are not compact. However, the circle, a finite interval, and a finite cyclic group are all compact.

We observe the compact-discrete theorem holds for all domains depicted in Figure 1 (which contains four of the six domains of signal processing of Steiglitz [6]). Infinitely extended discrete audio data (top left) is not compact (as it goes off to infinity), but is discrete. By the theorem, the frequency domain must be compact, which we observe being the case. The circle in the complex plane is compact. If we discretize the frequency domain by only allowing a discrete set, this automatically means that the time domain needs to be compact as well. This leads to both time and frequency domains of the DFT necessarily being compact, which we see in the bottom two domains of Figure 1. This is the topological reason why discrete Fourier transforms necessarily lead to circular convolutions.

In practice, one does not encounter truly infinite sequences of audio data. Instead, audio data are finite in length. This introduces a notion of a starting point and an end point of the linear domain, which differs from the infinitely extended non-compact discrete domain in that it is both finite and, due to the endpoints, has boundaries. It also differs from circular domains which do not have endpoints. We will say that a sample, an extremum, or any other localized aspect of audio data is *in the boundary* if it contains the first or the last sample of the finite linear domain under consideration. We will be dealing with finite linear domains; hence, all our linear domains will have two boundaries. In this paper, both linear and circular domains will be discussed and we will see that circular domains can be beneficial outside of considerations of discrete Fourier transforms.

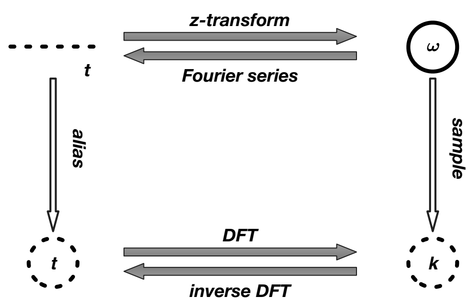


Figure 1: Discrete time domains of signal processing after Steiglitz [6]. Due to Pontryagin duality, topologically compact domains are dual to discrete domains and vice versa. Hence if both domains are discrete, they both also necessarily need to be compact. This leads to the well known property of the Discrete Fourier Transform necessarily involving circular convolution.

[2] This particular notion of window is not to be confused with familiar notion of a window in digital signal processing [11]. They have different definitions and serve unrelated purposes.

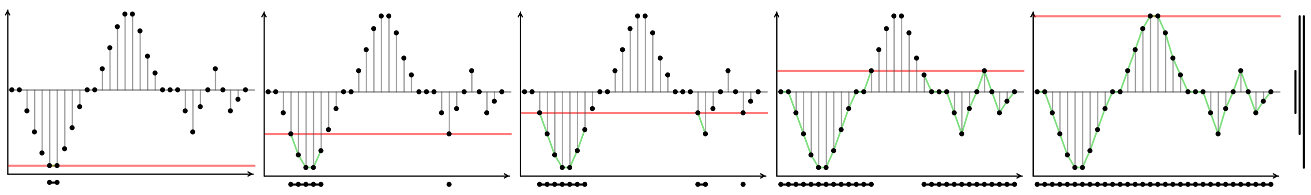


Figure 2: Sublevel set persistent homology of a finite audio signal. As we raise the level line, more of the signal is included. The connected component at each level (red) is depicted below the time series graph. At extrema, connected components are created or merged.

4. SUBLEVEL SET PERSISTENCE

We rapidly review sublevel set persistence of discrete audio data. For more details see [1, 9] [3] . Sublevel set persistent homology computes the changes in connectivity as a level set moves across the amplitude of discrete audio data. The idea is depicted in Figure 2 [4] . The figure shows a red horizontal line. This line marks a level. A sublevel set is a subset of the original sequence of samples such that a sample is included if and only if its value is at or below the level set.

We visualize a current sublevel set by a horizontal line-graph that is depicted below the x-axis as copies of the included sample without amplitude, but with their neighborhood relationships depicted by connecting lines.

As the level moves from below the global minimum to above the global maximum, the sublevel set changes. Steps of this process are illustrated in Figure 2. Each subfigure illustrates what happens when a local minimum or maximum is reached. Observe that mimima create a new connected component in the sublevel set, while maxima, that are not in the boundary, merge connected components. The general insight that topology (here with respect to number of connected components) changes is core to Morse theory as developed by Marston Morse [25] in differentiable topology. The idea has been translated into computational settings in the context of the development of computational and applied topology [26]. Morse theory is the core idea that allows us to process digital audio data, even if it breaks requirements of the original theory such as isolated extrema and the ability to always define a non-zero gradient. As shown in [1, 9], the restrictions of classical Morse theory can indeed be removed without issue for discrete sequential data, allowing us to use this theory in the setting of discrete audio samples.

We will use the so-called *barcode* [27] as representations of topological change. The barcode is a set of vertical bars on the right of Figure 2. It keeps track when a connected component is created at a minimum. This is the bottom point of the bar. The bar extends to the level of a maximum where a connected component was merged with another, hence the connected component associated seizes to continue independently. Hence, each bar corresponds to one minimum and one maximum. The global maximum has a special status in that it connects all connected components that are still separate into one. Technically, one bar in the barcode is *essential* [28] in that it describes the final connected component. We will not use that property here and simply have the essential barcode end at the global maximum. In fact, barcodes are not the primary object of this paper. We are primarily concerned with manipulations such that barcodes are not altered. They are a means to visualize the topological invariants that we use to segment regions of deformability. We will discuss their behavior under streaming updates of audio data in Section 10.

[3] Topology in digital audio is reviewed and motivated in greater detail in tutorial lectures given at DAFx2022 and DAFx2023 [8, 24].

[4] The figure is based on Figure 1 in [1], however, correcting an error in the barcode.

5. BOX SNAKES

A *box snake* is a segmentation of a finite block of audio data into regions of extrema, and regions of monotone sequences between them [1]. Box snake will refer to the whole structure, while we will call individual segments either a *snake box* or just a *box*. Each box consists of a start and end sample, the range of amplitudes permissible inside the box. The name box snake is an extension of the term *snake* introduced when studying alternating extrema [29]. Alternating extrema are natural in our setting because for one-dimensional audio data a minimum necessarily needs to follow a maximum and vice versa. Box snakes capture more information because the monotones between extrema are also represented in the data structure. Directionality differentiates between individual boxes. Monotones can be either ascending or descending, and extrema can be either minima or maxima. It will be convenient to be able to identify if two extrema are the same without specifying if it is a minimum or maximum. For this purpose we refer to "extrema of a given *type*", where type can be either an minimum or maximum. There are a wide range of duality results for extrema and monotones in digital audio [9], hence for many arguments, it is more important to know if the extremum is the same or differs rather than what specific type of extremum it is.

Sublevel set persistent homology can only change at local minima and local maxima, with the caveat that maxima which are in the boundary of a linear domain do not merge connected components. This means that the box snake structure segments audio data into areas that correspond to potential topological change (extrema) and areas that do not change the topology (monotone sequences and maxima in the boundary). As long as monotonicity is maintained, the boxes corresponding to monotone sequences can be modified without changing the topology captured by the sublevel set persistent homology [1, 9]. Monotone deformations were realized using piece-wise linear monotonic interpolations [1], though any monotonic deformation is permissible. An example of a box snake for both linear and circular audio data is shown in Figure 4. These two domain types can be related to each other via surgery.

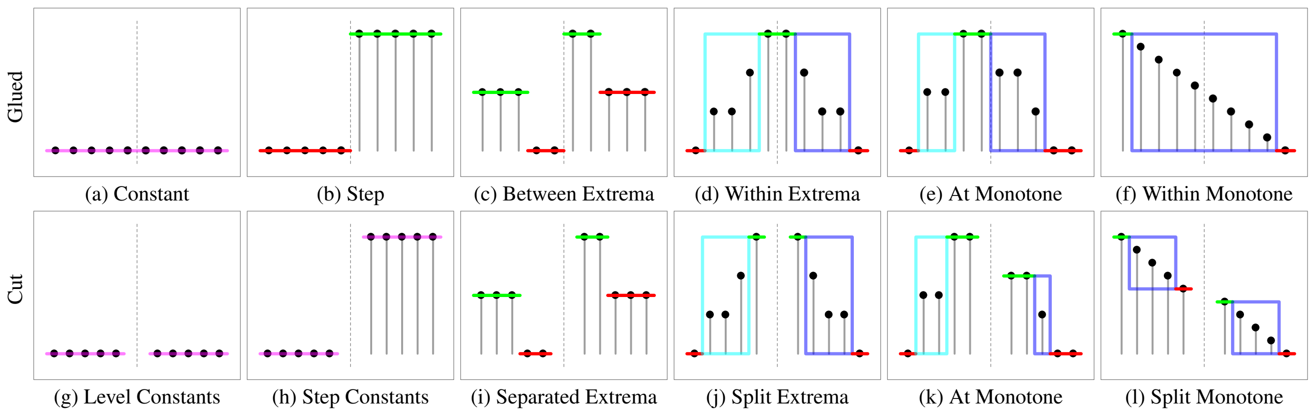


Figure 3: Classification of surgery changes box snakes. Top row (a)-(f) shows glued cases, and bottom row (g)-(l) shows cut cases. Colors indicate the type of box snake. Purple is a constant (both minimum and maximum), red is a minimum, green is a maximum, cyan is an ascending monotone, blue is a descending monotone. The dashed vertical line indicates the surgery position.

6. SURGERY OF BOX SNAKES

Topologically, one can think of changes to a domain as surgical moves. Specifically, we can describe changes that sever connectivity as a *cut* and those that create connectivity as *gluing*. In our settings, we are talking about splitting a larger sequence of audio samples into smaller ones, or concatenating smaller sequences into a larger one. Finally, we are also interested in taking a sequence of a given length and gluing it into a circle by connecting the ends of the sequence. This leaves the content and length of the sequence unchanged but changes its topology. Topologically, a cut introduces two new boundaries. One on each side of the cut. A gluing operation undoes this cut by connecting these two boundaries.

Surgery on the domain also alters the associated box snake decompositions. All possible effects of surgery on the snake box structure are enumerated in Figure 3 with the top row showing the glued condition for the matching cut condition in the bottom row below it. Two general cases are cutting between existing boxes and cutting within a box. If a cut happens between two existing boxes, there are two basic cases. (1) If the boxes are both extrema, then there is no need to alter the box snakes (Figure 3(c) and 3(i)) unless it happens to be a global step function (Figure 3(b)) in which case both sides of the step become constant functions (Figure 3(h)). (2) If one of the boxes is a monotone (Figure 3(e)), then the monotone box needs to be modified (Figure 3(k)) using the following rules: The flat next to the cut becomes a box of the appropriate new boundary extremum, while the remainder of the former monotone (if there is one) is now a monotone shortened by the samples that now form the boundary extremum. If the surgery occurs within a snake box, we are guaranteed to end up with two boxes that capture the two new boundary extrema (Figures 3(j) and 3(l)). In the case of splitting a monotone, there may also be reduced size monotones on one or both sides (Figure 3(l)), depending if there are audio sample sequences left. In the case of within a flat extremum (Figures 3(d)), one ends up with the extremum split into two boundary extrema of the same type (Figures 3(j)). The rule also applies to cutting a constant function (Figure 3(a) and 3(g)).

Observe that the number of updates to the snake box structure is constant with the number of snake boxes involved in all cases. In some cases, no new box needs to be created (Figures 3(b)(h), 3(d)(j)) to the case where one box is replaced by at most four (Figure 3(f)(l)).

In this paper, we will use these rules for two purposes: implement shifts in data while updating the box snake structure, and relating linear and circular domains via surgery.

7. LINEAR AND CIRCULAR DOMAINS OF AN AUDIO WINDOW

Box snakes can be computed for both linear and circular domains. Doing so illustrates that there are specific effects that differ between these two domain types. The circular domain is characterized by having no boundary. A linear domain is characterized by having a first and a last sample in the domain. Those two samples form the boundary of the linear domain. Both types of domain are related by a single surgery. Gluing the boundaries of a linear domain, by making the predecessor of the first sample the last sample, and the successor of the last sample the first, one arrives at a circular domain with identical samples in order. If one cuts between adjacent samples of a circular domain by removing the successor of one and the predecessor of the other, one arrives at a linear domain. If the adjacent

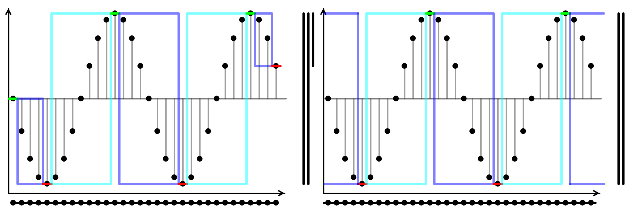


Figure 4: Box snake segmentation for a linear (left) and a circular (right) domain of a sinusoid. The linear domain creates boundary extrema, while the circular segmentation creates a phase-independent correct segmentation of the sinusoid. Notice that the boundary effects in this case also creates an extra barcode. It can be viewed as an artifact of broken periodicity. Box colors: increasing monotone (cyan), decreasing monotone (blue), minima (red), and maxima (green).

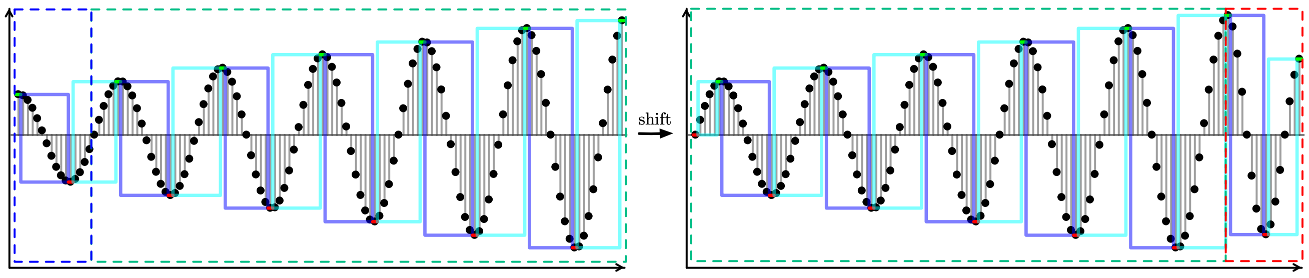


Figure 5: A linear left-shift of a 128 sample sequence by a block of 16. First the original sequence is cut on the left and the left side (dashed blue) is discarded. Then the remaining piece (dashed green) is glued on the right with a new sequence of three samples (dashed red). Solid boxes are box snake segments. Notice that the shift requires a surgery of a monotone on the left, and a glue next to an extremum on the right. To make this shift circular, it first needs to cut the circularity, perform the shift, then glue the boundaries.

samples are the same as those used in the gluing described above, this is the inverse process; otherwise, the samples in the linear domain will be cyclically shifted depending on the choice of the cut location. Using the associated box snake surgery update from Figure 3 allows one to relate box snakes between these two cases.

Topologically, the boundaries of the linear domain introduce effects. Figure 4, which shows the box snake segmentation of a discrete sinusoid assuming a linear domain (left) and a circular domain (right). The linear case as boundary extrema that do not relate to the periodic structure of the sinusoid and what should be one monotone slope through the boundary has been split by boundary extrema into two separate monotonic segments. The circular case cleanly captures the extremal and monotone structure of the sinusoid without these boundary artifacts. This effect then carries over to topological analysis methods such as the sublevel set persistence. Given that the right boundary extremum of the linear domain is a minimum, it will create a connected component, and hence a bar in the barcode. This bar is an artifact of the linear domain (left) and is absent from the circular barcode (right). Thus, there is an argument that if a finite array is related to periodic data, circular domains are preferable over linear domains.

As we have already seen, there is an immediate relationship between the linear and circular topology of a domain of audio data via a single cut/glue operation. Any further cuts of a linear domain will produce two linear domains. Hence, any subsequence of a circular domain, which can be extracted surgically via two cuts, is again a linear domain.

Given that any subsequence of a circular domain is always linear, and of course any subsequence of a linear domain is linear, it is possible to always relate subsequences even if there is a mismatch in the topology of the total domain. This is the core property used to connect the two concepts, which we will utilize in connecting circular buffers as can be useful for DFT-type computations, and linear time-domain buffers used for streaming audio to the DAC for playback. Even if one arrives at a linear subsequence, one can in principle always "circularize" the sequence by a single glue operation.

8. REALIZATION AND EFFICIENCY OF SHIFTS VIA SURGERY

Streaming audio depends on finite buffers, which are updated based on new audio data arriving. Taking a finite total buffer length N, and an update of audio data of length L that is shorter than the total buffer length. This amounts to an L-shift. An example of the shift to the left of a buffer of total length $N = 128$ and a shift length $L = 16$ is depicted in Figure 5. The left side of the figure shows the data before the shift, and the right side of the figure shows the data after the shift. The blue dashed rectangle indicates the block of audio samples to be removed due to the shift, while the red dashed rectangle indicates the new data introduced by the shift. If the length of the updated audio is close to the total length of the buffer, it may be most efficient to simply recompute the box snakes for the whole buffer after each shift. However, if the shifted data is substantially shorter, it becomes efficient to compute the box snake structure only for the new data, and update the changes to the box snake structure via the surgery moves discussed in section 6. Given that surgeries only need to touch a constant number of boxes, sublinear performance relative to the full length of the audio sequence can be achieved. In the example of Figure 5, the ratio of total buffer length to shift length is $8 : 1$. Hence, one should expect a performance gain close to this ratio by performing surgery over brute force. If the buffer is circular, one additional cut before and glue after need to be performed, at an additional constant cost relative to sample buffer size.

9. PRACTICAL BUFFER-BASE ARCHITECTURES PIPELINE

Some forms of digital audio processing involve buffer sizes. One such buffer size is usually chosen as part of audio playback. The number of audio samples to be fed to the audio hardware is fixed, usually to a power-of-two number. This is then reflected in the choices of APIs defined for audio rendering. For example, the current WebAudio standard defines an audio output buffer size - called render quantum - of 128 [30, 31]. For audio playback, it is desirable to keep buffer sizes small to minimize or avoid perceptible delay. While the specific details of time-delay effects on perception are complicated [23], it is generally accepted that delays less than 3 to 10 ms is either imperceptible or non-disruptive. This leads to buffer sizes of 128

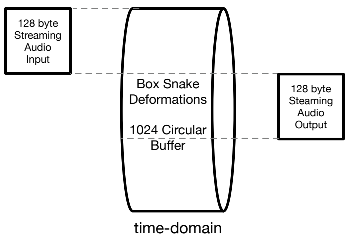

128 byte Streaming Audio Input
Box Snake Deformations
1024 Circular Buffer
128 byte Steaming Audio Output
time-domain

Figure 6: Short linear audio buffers in a pipeline that allows for box snake deformations in both time and frequency domains, and accommodates double buffering.

or 256 being deemed sufficient at 44.1 to 48 kHz sampling rates in many practical application settings.

Other audio processing strategies can also involve their own buffer or window sizes. For example, the most widely used form of the Fast Fourier Transform also requires a size that is powers-of-two [6]. However, the motivation behind this specific buffer size differs from the one driving the size of buffers for audio playback. For the Fourier transform, a larger window size leads to a finer frequency resolution [6]. Although, the desired resolution is application dependent, 1024 to 8192 size buffers are frequently encountered for many real-time applications, both smaller and larger sizes are in use. Web Audio supports FFT sizes of 32 to 32768 [30].

Furthermore, different buffering mechanisms may have different underlying topological considerations. Streaming audio inhabits a sequential, "linear" domain, while the Discrete Fourier Transform (and hence the FFT) requires a periodic (cyclically closed) domain. A classic example of bridging both linear and circular domains is the case when using the FFT for fast linear convolution via methods such as overlap-add and overlap-save [5, 32]. However, even if no FFT is involved, it can be beneficial to utilize a circular buffer as a data structure of fixed length in signal processing.

9.1 Example 1: Streaming Time-Domain Architecture

A simple architecture for time-domain deformations using box snakes is depicted in Figure 6. It shows real-time incoming audio, which could be a microphone, streaming audio media, or generative audio, here depicted with a length of 128 samples and a larger circular buffer (here depicted with a length of 1024 samples) on which we will compute the box snake structure and allow time-domain audio deformations. Playback is achieved by copying a subsequence into the audio output at the appropriate size. The readout of the buffer for audio-playback does not require any surgery. The depiction shows an offset between input and output reflecting a double-buffering-type strategy.

Notice that the box snake structure can be initialized at minimal computational cost if one starts from silence. The buffer is initialized with a constant function (silence) for which the box snake structure can be constructed without inspecting the samples at constant cost independent of the size of the buffer.

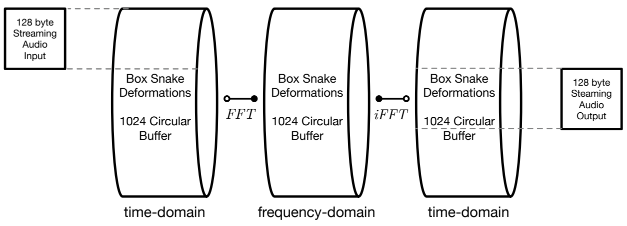


Figure 7: Short linear audio buffers in a pipeline that allows for box snake deformations in both time and frequency domains, and accommodates double buffering.

9.2 Example 2: Streaming Time/Frequency-Domain Architecture

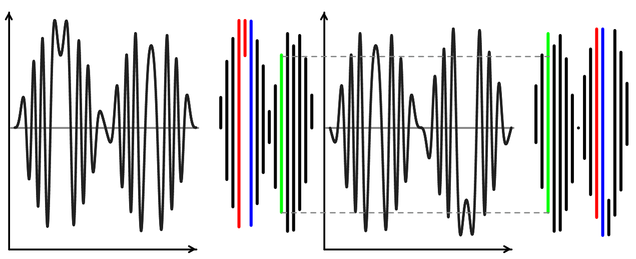
The simple structure of Figure 6 can be extended as needed. For example, it may be desirable to deform the audio data in the frequency domain. An architecture that allows frequency-domain manipulation as well as time-domain manipulations before and after the transforms is depicted in Figure 7.

The architecture uses an audio-IO buffer size of 128 samples and an FFT size of 1024 reflecting an $8 : 1$ ratio matching the one illustrated in Figure 5. Instead of one circular buffer of the time-domain case described in the previous section, this architecture consists of three circular buffers. Hence, insertion cost will only ever be of the order of the audio-IO size (in the illustrated example 128), given that the cost of the surgery step to prepare the first circular buffer is $O(0)$. The box snake structure can then be used to deform the time series [1]. We then transform the circular buffer to construct a frequency domain representation of the audio of the given length. For purposes of deformation, the box snake structure is computed over the whole domain. Given that the spectral changes can be (and in general are) global, the complete box snake structure has to be recomputed at each shift, hence requiring linear computation cost. This is still below the transform cost of the FFT of $O(N \log N)$. The inverse FFT (iFFT) then returns the deformed frequency data ($O(N \log N)$ back into time-domain signal in which one could decide to allow a further time-domain deformation, requiring another global snake box construction. Observe that deformations in the frequency domain are substantially more expensive than the pure time-domain case (which is sublinear), but the extra cost is dominated by the cost of computing the FFT and its inverse at $O(N \log N)$. The topology and snake box construction match the circular topology imposed by the FFT.

10. BARCODE CONSTRUCTION FOR STREAMING

Barcodes capture the topological invariant around which we deform our audio signals. Hence, ease of interpretability and coherence with progressing data are of great importance. Shifts remove some minima and maxima while introducing new ones. Hence, the barcode representation of sublevel set persistent homology needs to be dynamically updated in shifting applications.

Barcodes denote the creation of connected components with the heights of minima, and the death of them by the


(a) Linear Domain

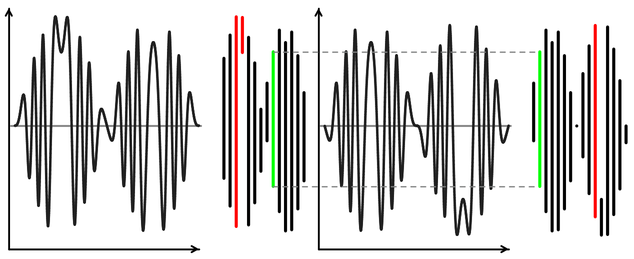

(b) Circular Domain

Figure 8: Barcodes from local shift rules for (a) linear and (b) circular domains. Red bars reaching the global maximum, blue is a bar at the global maximum due to the topology being linear. Green is an example of a bar that had to be reconfigured in the linear case but remains unaltered in the circular case. The global maxima forces a local orientation change in the bar code rule in the linear case, forcing the reconfiguration of bar codes when the global maximum changes due to shifts.

heights of maxima. However, given that maxima merge connected components, there will be one connected component per maxima that continues, while all other connected component ends at that maximum. The decision to resolve which prior connected components merged at a maximum continues needs to be resolved by a rule that we will call *barcode construction rule*[5] As we have already seen, some barcode construction rules like the elder rule can span the full length of an audio sequence and lead to frequent updates of the barcode. This creates erratic barcode updates that are not clearly related to any pertinent information for audio analysis.

This problem can be overcome by using a circular topology. Due to the absence of a boundary, it is easy to see that audio data on a circular topology are either constant or have for each minimum an opposite maximum both to its left right (and vice versa). Finally, because minima and maxima necessarily need to alternate, we get that we have the same number of each. Hence, we can always associate a minimum with a neighboring maximum, either to the left or to the right. And there is no exception for maxima in the boundary because there is no boundary. Given that all minima create a connected component, and in this case all maxima merge connected components, we are done. A local strategy of constructing barcodes is possible in the circular case.

Attempts at constructing local rules for the linear case reveal that they have to reorient at one global maximum (flip from right to left, or vice versa) [9]. This means that if the global maximum changes, such as due to being shifted out or a new global maximum being shifted in (or both), the local direction in which minima and maxima are connected will have to change. There is no way to achieve non-altering behavior of barcodes in general on linear domains. This effect is illustrated in Figure 8. Each case shows a shift by half the total sample size from the left to the right. The left case 8(a) shows the linear domain. The effect is that at a global maximum connected components from both the left and right are necessarily merged because they are separated by boundaries. A local barcode construction only connects neighboring extrema. Due to the above effect the immediate neighbors of the global maximum requires barcodes from the immediate minima to the left and right. Any subsequent barcode must use the next maximum and merge with the unused minimum next to it. Hence on the left side of the global maximum we max-min pairs grow leftwards, while on the right side of the global maximum the max-min pairs grow rightward. When the global maximum changes, it requires that all barcodes are constructed as above, which can be flip between leftward pairing to rightward pairing.This effect can be seen on the example of the bar color coded in green.

In the circular case, 8(b) this phenomenon does not exist. At a global maximum, in the circular case, the last merge will close the domain with itself, hence instead of producing an extra bar, there is just one capturing the final closure. Thus, we have one less bar reaching the global maximum in the circular case. Notice also that there is nothing that forces reorientation. Hence, the green bar in this case remains unaltered. For this reason, it is recommended to use circular domains to achieve locally temporally coherent barcodes in shifts.

11. CONCLUSIONS

In this paper, we showed how existing audio deformation techniques based on sublevel set persistent homology [1] can be extended for streaming audio applications. The central strategy involves the use of surgery (cuts, glues) on the box snake structure to realize finite length shifts. We showed that linear buffers exhibit artifacting with respect to periodic data in the box snake decomposition due to effects of the boundary. On the other hand, circular domains are better behaved since they avoid boundary artifacts. Additionally, circular domains also arise as the topology of the Discrete Fourier Transforms (DFT), hence, making the joint use convenient. Basic processing architectures have been proposed that allow time and time/frequency-domain deformation of audio data at cost equivalent to that of two FFT transforms. Pure time-domain processing can be performed at linear or better cost. We also showed that circular topologies allow for temporally coherent behavior for barcode representations which track the topological changes in the audio signal.

Work on sublevel set persistence on digital audio is still largely unexplored. For example, little is known about the precise relationship of sublevel set persistent homology when time and frequency-domains are considered together. The techniques discussed here and in [1] can form the basis for audio analysis, which also has not been explored in any detail yet. These are areas for future work.

[5] This topic is covered more extensively in [9]. Here we will limit ourselves to streaming applications.

12. REFERENCES

[1] G. Essl, "Topology-Preserving Deformations of Digital Audio," in *Proceedings of the International Conference on Digital Audio Effects (DAFX-24)*, Guildford, UK, 2024, pp. 329–336.

[2] ——, "Topologizing Sound Synthesis via Sheaves," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2021, pp. 260–267.

[3] ——, "Deforming the Oscillator: Iterative phases over parametrizable closed paths," in *Proceedings of the International Conference on Digital Audio Effects (DAFX-22)*, 2022.

[4] P. Bendich, H. Edelsbrunner, D. Morozov, and A. Patel, "Homology and Robustness of Level and Interlevel Sets," *Homology, Homotopy and Applications*, vol. 15, no. 1, pp. 51–72, 2013.

[5] T. G. Stockham Jr, "High-speed convolution and correlation," in *Proceedings of the April 26-28, 1966, Spring joint computer conference*, 1966, pp. 229–233.

[6] K. Steiglitz, *A Digital Signal Processing Primer, with Applications to Digital Audio and Computer Music*. Addison Wesley, 1997.

[7] P. Kenny, "Pontrjagin duality and digital signal processing," *IEEE Transactions on Education*, vol. 38, no. 2, pp. 131–135, 1995.

[8] G. Essl, "Topology in Sound Synthesis and Digital Signal Processing–DAFx2022 Lecture Notes," *arXiv preprint arXiv:2211.05821*, 2022.

[9] R. Belton and G. Essl, "Discrete Level Set Persistence for Finite Discrete Functions," 2025, submitted to Foundations of Data Science. arXiv preprint https://arxiv.org/abs/2501.17794.

[10] S. Cultrera di Montesano, H. Edelsbrunner, M. Henzinger, and L. Ost, "Dynamically Maintaining the Persistent Homology of Time Series," in *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2024, pp. 243–295.

[11] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[12] R. Biswas, S. Cultrera di Montesano, H. Edelsbrunner, and M. Saghafian, "Geometric characterization of the persistence of 1D maps," *Journal of Applied and Computational Topology*, pp. 1–19, 2023.

[13] M. Robinson, *Topological Signal Processing*. Springer, 2014.

[14] A. Ortega, *Introduction to graph signal processing*. Cambridge University Press, 2022.

[15] G. Essl, "Topological IIR filters over simplicial topologies via sheaves," *IEEE Signal Processing Letters*, vol. 27, pp. 1215–1219, 2020.

[16] ——, "Causal Linear Topological Filters Over A 2-Simplex," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8842–8846.

[17] J. Araujo-Simon, "Compositional nonlinear audio signal processing with Volterra series," *arXiv preprint arXiv:2308.07229*, 2023.

[18] D. V. Nort, M. M. Wanderley, and P. Depalle, "Mapping Control Structures for Sound Synthesis: Functional and Topological Perspectives," *Computer Music Journal*, vol. 38, no. 3, pp. 6–22, 2014.

[19] S. Kasich, "Possible applications of knots in Computer Music and NIMEs," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2024, pp. 198–205.

[20] J. A. Perea, "Topological Time Series Analysis," *Notices of the American Mathematical Society*, vol. 66, no. 5, pp. 686–694, 2019.

[21] N. Sanderson, E. Shugerman, S. Molnar, J. D. Meiss, and E. Bradley, "Computational topology techniques for characterizing time-series data," in *Advances in Intelligent Data Analysis 16*, 2017, pp. 284–296.

[22] S. Emrani, T. Gentimis, and H. Krim, "Persistent Homology of Delay Embeddings and its Application to Wheeze Detection," *IEEE Signal Processing Letters*, vol. 21, no. 4, pp. 459–463, 2014.

[23] B. C. J. Moore, *An Introduction to the Psychology of Hearing*, 6th ed. Brill, 2012.

[24] G. Essl, "Combinatorial Hodge Theory in Simplicial Signal Processing – DAFx2023 Lecture Notes," *arXiv preprint arXiv:2311.03469*, 2023.

[25] J. W. Milnor, *Morse theory*. Princeton University Press, 1963.

[26] H. Edelsbrunner and J. L. Harer, *Computational topology: An introduction*. American Mathematical Society, 2010.

[27] R. W. Ghrist, *Elementary Applied Topology*. Createspace, 2014.

[28] T. K. Dey and Y. Wang, *Computational Topology for Data Analysis*. Cambridge University Press, 2022.

[29] V. I. Arnol'd, "The calculus of snakes and the combinatorics of Bernoulli, Euler and Springer numbers of Coxeter groups," *Russian Mathematical Surveys*, vol. 47, no. 1, p. 1, 1992.

[30] J. Reiss, *Working with the Web Audio API*. Focal Press, 2022.

[31] H. Choi and P. Adenot, "Web Audio API 1.1," W3C, W3C Working Draft, Nov. 2024, https://www.w3.org/TR/2024/WD-webaudio-1.1-20241105/.

[32] C. Burrus, "Block realization of digital filters," *IEEE Transactions on Audio and Electroacoustics*, vol. 20, no. 4, pp. 230–235, 1972.