FEATURE-BASED DELAY LINE USING REAL-TIME CONCATENATIVE SYNTHESIS

Niccolo Abate and Brian Hansen

Department of Computational Media
University of California, Santa Cruz
Santa Cruz, California, USA
nlabate@ucsc.edu | brmhanse@ucsc.edu

ABSTRACT

In this paper we introduce a novel approach utilizing real-time concatenative synthesis to produce a Feature-Based Delay Line (FBDL). Expanding upon the concept of a traditional delay, its most basic function is familiar – a dry signal is copied to an audio buffer whose read position is time shifted producing a delayed or "wet" signal that is then remixed with the dry. In our implementation, however, the traditionally unaltered wet signal is modified such that the audio delay buffer is segmented and concatenated according to specific audio features. Specifically, the input audio is analyzed and segmented as it is written to the delay buffer, where delayed segments are matched to a target feature set, such that the most similar segments are selected to constitute the wet signal of the delay. Targeting methods, either manual or automated, can be used to explore the feature space of the delay line buffer based on dry signal feature information and relevant targeting parameters, such as delay time. This paper will outline our process, detailing important requirements such as targeting and considerations for feature extraction and concatenation synthesis, as well as discussing use cases, performance evaluation, and commentary on the potential of advances to digital delay lines.

1. INTRODUCTION

1.1. Concatenative Synthesis

Concatenation synthesis, thought of as directed granular synthesis [1] and referred to as its natural successor, is a type of synthesis where small segments of audio are selected according to their descriptors and concatenated together to create a unique audio stream. The origins of the idea are espoused in Iannis Xenakis' Formalized Music in which he describes a stochastic approach of concatenating small segments of audio together to create new sounds [2]. Due to processing limitations of the time, the analysis was a time consuming endeavor that had to take place pre-synthesis and was thus a major limiting factor for the synthesis technique. Concatenation synthesis gained prominence in vocal synthesis in the 1990s through work presented by various researchers including Hunt and Black in 1996 [3] and J Olive in 1997 [4]. The technique was quickly seen to be effective for resynthesizing sounds from a corpus of source audio comprised of vocal sonic components such as phonemes, sibilants, or fricatives. Concatenation synthesis remains as one of the predominant means of performing vocal synthesis today.

The early 2000s, referred to as the early years of concatenation synthesis [5], presented an increase in processing power spawning a surge of new interest in the technique exhibiting several different applications with an exploration towards musical ends. Ari Lazier and Perry Cook developed Mosevius [6], a tool for creating "audio mosaics." This application was available as a standalone tool or as a library, where it allowed users to perform concatenative synthesis on a corpus of audio using MIDI or real-time feature extraction on a control signal to inform segment selection. Similarly, Diemo Schwarz' CataRT [7] was a collection of patches for Max/MSP built to perform concatenative synthesis. This system also used manual control or real-time feature extraction of input audio to inform segment selection from a corpus of audio. These two applications were nearly identical in their approach to concatenative synthesis and established a standard paradigm for the process.

More recently, advancements in processing and research have led to the development of more tools utilizing concatenation synthesis in various ways. In 2011, Beller [8] created a physical gestural controller to control segment selection in a concatenative speech synthesis system for performance at IRCAM, which was later built upon by Zbyszyński et al. [9] in 2019, bringing the idea to musical ends along with the introduction of machine learning algorithms. In 2012, S. An et al. created a framework in which plausible accompanying audio is generated for physics based cloth animations by producing a simple target sample based upon simulation information which informs concatenation on a database of higher fidelity cloth samples [10]. In 2016, the audio plug-in Mosaic [11] brought concatenative synthesis towards the audio effects world by layering sounds from an audio corpus on top of incoming audio by means of real-time feature extraction of the input signal guided by user-specified thresholds for particular features. In 2017, MIT produced an application called RhythmCAT [12] that used concatenation synthesis to power a drum programmer and beat maker for electronic music. The application had a focus on refined interface and streamlined functionality, taking advantage of methods like dimension reduction and onset detection to quantize the output and smooth out the user experience. In 2019, C Moore and W Brent explored concatenation synthesis with a new level of interactivity using virtual reality technology to allow users to explore the feature space in three dimensions while forming clustering structures and allowing the user to explore the space with rays and other physical means [13].

While each of these applications explored concatenation synthesis in unique ways, they all rely on a corpus of audio that is pre-analyzed, and they appeal to the design paradigm established in the 2000s by apps such as Mosevius and CataRT. The reason for this is clear, as it is computationally efficient to pre-analyze an audio corpus and then use the resulting meta-data analysis to perform real-time synthesis. However, with current advances in processing and research in the field, we can expand this paradigm and tap into unexplored territory via generating audio corpora and performing end-to-end concatenation synthesis all in real-time. This may allow the technique to proliferate in the audio effects world, where relatively little work has been done but more potential exists for the technique in audio production and sound design.

1.2. The Delay Line

The phenomenon of audio delay fundamentally influences our auditory experience in the physical world. Acoustical sound waves propagate throughout a space and reflect off surfaces causing the superposition of time-offset waves at the position of a listener's ear or a microphone. Depending on the delay time, this phenomenon yields changes to the perceived audio ranging from an audible echo to intricate alterations in audio timbre resulting from spectral filtering. Notably, the perception of the quality of a space, such as a concert hall, is an amalgamation of all delays resulting from sonic reflections propagating throughout that space [14].

Given this correlation, it comes as no surprise that delay, harnessed through "delay lines," constitutes a crucial aspect of signal processing, whether analog or digital. Delay lines form the foundation for numerous common operations such as filtering, reverb emulation, and physical modeling, as well as various other audio effects like flanging and chorus [15].

With such a fundamental role in audio processing, advances in delay line technology have the potential to permeate multiple areas of signal processing and audio effects. As a result, the delay line represents a significant object of inquiry for modern processing power and algorithms, including real-time music information retrieval and concatenative synthesis.

2. FEATURE-BASED DELAY LINE

This brings us to our proposed model of a Feature-Based Delay Line (FBDL), a novel application of a traditional delay line that utilizes concatenation synthesis where the corpus of audio exists as an ever-changing delay line buffer, analyzed, segmented, and concatenated all in real-time (Figure 1). To accomplish this, segmentation and feature extraction take place as the audio is copied to the delay buffer, where the resulting analysis remains paired with its associated audio as it travels through the buffer. Segments are then selected according to a process we call targeting and concatenated to create the wet signal, which is then mixed with the dry signal. To control concatention of the delay line, the user can set several important parameters, including segment size, feature set, feature weights, targeting method, and targeting parameters. In particular, the critical method of targeting produces a system of selection criteria that expands on the concatenation synthesis norm of simple matching, adding tunable depth to the system and creating the delay-like behavior. The aggregate result is an audio effect that expands on the traditional delay line, exhibiting creative potential for audio production and sound design, as well as potential as a component for signal processing.

As discussed above, the fundamental structure of the FBDL is a delay line, where the content of the delay line buffer constitutes the audio corpus for concatenation synthesis. Our approach is rooted in the functionality of a traditional digital delay line [15], but it expands on the capabilities of the traditional paradigm to create new possibilities. A traditional digital delay line utilizes a circular buffer, where given an audio buffer X of sample size N, the next input sample will write to an indexed buffer position n, such that X(n) = input sample. As audio samples are written, the index n will then be incremented, wrapping around when it reaches the end of the buffer. In a digital delay line, given a delay time t in samples, the delayed signal will read every sample from index n - t (this index will also wrap to stay in the bounds of the buffer), such that the output = X(n - t). Our FBDL is similar to the traditional paradigm, where incoming audio is written to a circular buffer of length N. However, in our case the delay time, now denoted c, is dynamically determined as a result of concatenative synthesis, where the delay buffer position n - c is a function of audio features defined via the targeting process (Figure 2).

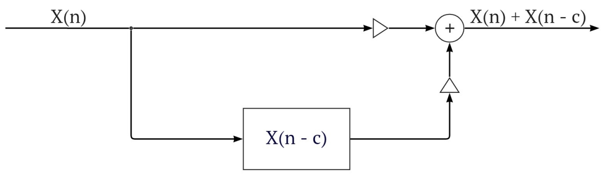

Figure 2: *Feature-Based Delay Line, where delay time c is determined by audio feature analysis (targeting).*

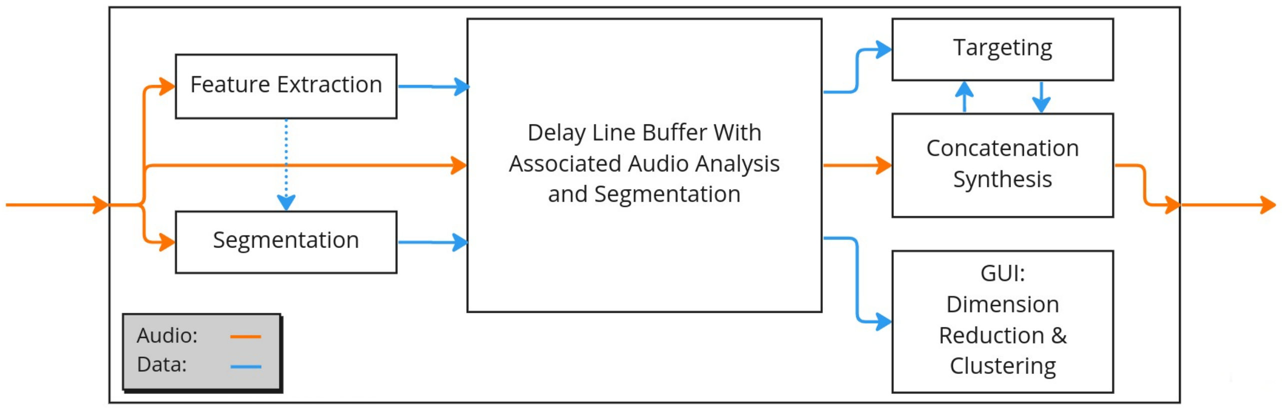
Feature Extraction
Segmentation
Delay Line Buffer With Associated Audio Analysis and Segmentation
Targeting
Concatenation Synthesis
GUI: Dimension Reduction & Clustering
Audio:
Data:

Figure 1: *Data and signal flow of the Feature-Based Delay Line Architecture.*

3. TARGETING

Targeting is our process of traversing a defined feature space and selecting audio segments according to their qualitative position in the space. The method of targeting plays the primary role in determining the delay position within our delay buffer, serving as the bridge between the concatenative synthesis technique and the delay line buffer. All audio that exists inside the delay buffer has been segmented in time and analyzed such that each segment of audio has an associated feature set of descriptors positioning it within a multidimensional feature space. Segments organize in the space such that qualitatively similar audio will be positioned close together and selected accordingly.

For audio selection, a target position in the feature space and a radius about the position are specified. At any given point in time, audio will be selected with a position in feature space contained within the target radius. We define the targeting method as follows:

$$T_c = Targeting(T_{ref}, R) \tag{1}$$

where T_c is the delay time in buffer position output from targeting (equivalent to c in the previous Figure 2), T_{ref} is the reference delay time used to determine the target position in feature space, and R is the target radius about the target position.

Inside the targeting function, given T_{ref} and R, the analysis of each segment of audio in the delay line is compared to the analysis of the audio segment containing X(n - T_{ref}), the "reference segment," where n is the current write position in the delay buffer.

Formally, all audio segments in the buffer can be stated as:

$$S_i = X(n - T_i), \cdots, X(n - T_i + l) \tag{2}$$

where S_i is the segmented audio starting at delay time T_i, with length l (Figure 3).

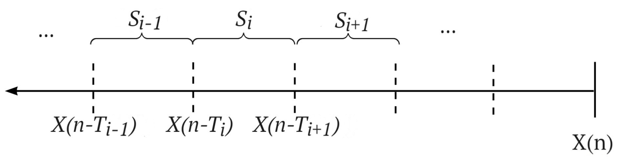

Figure 3: *Delay line segmented for targeting.*

Each audio segment, S_i, is associated with a feature set vector FS_i:

$$FS_i = (F_1, F_2, \cdots, F_m) \tag{3}$$

where F_m is a feature of the audio in segment S_i. The collection of all feature set vectors in the delay line forms our feature space.

The association between each S_i and FS_i is unique except in special cases. For example, given a feature set vector containing only RMS, two segments may share the same value. However, such an occurrence would be scarce and become progressively improbable as the dimensionality of the feature space increases. The only other case where this scenario may occur is with repetitive input signals, in which two segments contain identical audio. Even here, the scenario remains highly unlikely because it necessitates precise alignment with respect to segmentation and FFT framing.

The reference segment of audio, S_{ref}, is defined as the segment which contains the sample $X(n - T_{ref})$:

$$S_{ref} = S_k \mid X(n - T_{ref}) \in S_k \tag{4}$$

where T_{ref} is the target delay time, and S_k is the audio segment containing the sample $X(n - T_{ref})$ (Figure 4).

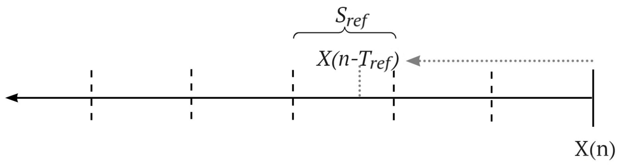

Figure 4: *Reference Segment S_{ref} determined by reference delay time Tref.*

Then, the Euclidean distance between the reference feature vector FS_{ref} and all other vectors in our feature space is computed. If the distance between a given feature space vector FS_i and the reference vector is less than the specified target radius, then the vector is stored in the set of viable segments V.

$$V = \{S_k \forall k \mid d(FS_k, FS_{ref}) \le R\} \tag{5}$$

A selection candidate S_{sel} is then randomly selected from V, and its reference time, T_{sel}, is output from the targeting function, ultimately setting the delay time T_c equal to T_{sel} (Figure 5).

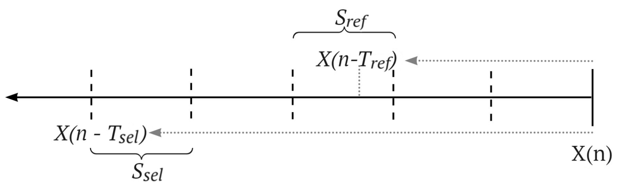

Figure 5: *Segment S_{sel} selected from the set of viable segments. S_{sel} must be sufficiently similar to reference segment S_{ref}. Associated time delay T_{sel} is set as c for the delay tap $X(n - c)$.*

The FBDL can be made to act like a traditional digital delay. For the targeting function detailed above, it is trivial to show how this is achieved. If we let R = 0 and assume the case that all FS_i are unique, then the only viable segment for which the euclidean distance between FS_i and FS_{ref} is less than or equal to R is the segment S_i = S_{ref}. Thus, the only viable segment is the reference segment, S_{ref} (Figure 6).

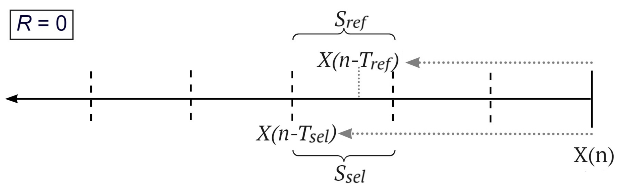

Figure 6: *Reference Segment S_{ref} is the only viable segment as Target Radius R = 0. S_{ref} therefore will be selected.*

However, there is a small discrepancy here: because T_{ref} might fall anywhere inside S_{ref}, there is some potential error from the exact reference delay time given depending on when the targeting method is queried. While this is not typically noticeable depending on the segment size used, it can be remedied with slight shifting of the selected segment. In this case, the selected segment will be shifted to align with the reference delay time if it is the reference segment containing T_{ref} (Figure 7).

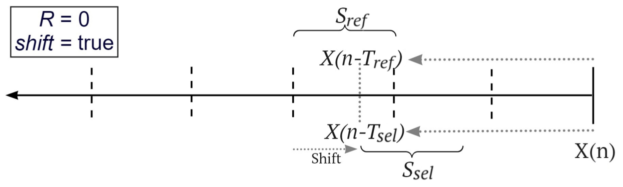

Figure 7: *Reference Segment S_{ref} is the only viable segment as Target Radius R = 0. With shifting enabled S_{sel} = S_{ref} is now shifted to align with T_{ref}.*

Now the output from the targeting method will always be T_{ref}. Therefore, in this scenario, the FBDL and the traditional digital delay line are identical, as $X(n - T_c)$ = $X(n - T_{ref})$.

As the value of R increases, more audio segments become viable candidates for selection, and thus more sonic variety is introduced into the delayed signal.

4. TARGETING EXPANSIONS - PARAMETERS AND METHODS

The targeting function can be modified or parametrically expanded to impact the behavior of the delay. This can be accomplished via adding parameters to a given targeting method or the targeting method itself can be modified to process input parameters in various ways. Our targeting function's expanded signature can be generalized as:

$$T_c = Targeting(T_{ref}, R, \cdots) \tag{6}$$

For our implementation we include the addition of feature weights and target smoothing, defining our final targeting function as follows:

$$T_c = Targeting(T_{ref}, R, FW, S) \tag{7}$$

where FW is a vector of feature weights, and S is a smoothing factor.

4.1. Feature Customization and Weighting

The feature set can be customized to alter the selection process and sound quality of the delay effect. Customization can result from a combination of features being added or removed from a given feature vector, or specific features in the vector may be replaced with others that are more desirable. The introduction of a feature weight vector allows the user to control the relative strength of each feature in the vector during the selection process, thereby accentuating specific audio characteristics in determining the viability of a segment. Computationally, this is a simple enhancement to the targeting function that can allow greater real-time control over the selection process. To accomplish this, compute the Hadamard Product [16], where given a feature vector and vector of feature weights both of the same length n:

$$FV_{wgt} = FV \circ FW = (F_i W_i, \cdots, F_n W_n) \tag{8}$$

where FV_{wgt} is the weighted feature vector, FV is the original feature vector, and FW is the vector of feature weights. The entry-wise product is computed for all vectors in the feature space before computing Euclidean distances and determining segment viability.

4.2. Smoothing

The smoothing parameter S changes the way the target position traverses the feature space by setting the feature set of the reference segment equal to the average of the feature sets of the prior N segments trailing the reference segment in the delay line. Thus, by applying a smoothing factor, the feature set of the reference segment is defined as:

$$FS_{smth} = \frac{1}{N}\sum_{i=0}^{N-1} FS_{ref-i}. \tag{9}$$

This effectively makes the size of the reference segment larger, as it incorporates the features of more audio into its average. As a result, the target may remain more centralized about the feature space and mitigate effects of outliers.

4.3. Targeting Methods

Targeting methods define distinct paradigms for traversing and organizing the feature space. Targeting methods are composed of unique targeting functions with varying parameter sets existing as arguments that fundamentally impact targeting behavior. Each targeting method may have a collection of parameters that apply to it that may or may not also apply to other targeting methods.

The primary targeting method utilized is a best fit approach, based on the audio input into the delay line, as well as a collection of other potential parameters such as target delay time, audio feature sets, feature weights, and smoothing, as detailed above. This method is parametrically automated, and as shown lends itself to unique potential for the FBDL.

Alternatively to our automated approach, manual targeting was also implemented. For this method the user manually determines the target position in feature space via use of a graphical user interface or other direct interactive methods. Our implementation focuses on the former, allowing a user to traverse and explore the feature space more freely and develop a deeper understanding of the feature space bounds. For example, it can be effectively used when writing to the delay line is paused, "freezing" the state of the delay line into a temporarily static corpus. With a static corpus and feature space, the listener can take time to consider how audio segments are associated with various qualities in the space. This can greatly inform the user on setting and refining parameters that impact the targeting process and segment selection.

Functionally, the behavior of the FBDL is critically determined by the targeting parameters and method. As shown above, with restrictive settings it will act exactly like a traditional delay line. However with modifications, the FBDL can produce a broad array of sonic behavior ranging from light variation to entirely new textures that are generated and layered into the original audio. In this way, our FBDL encompasses the full scope that a traditional delay line affords while introducing a broad set of new possibilities.

5. FEATURE EXTRACTION AND CONCATENATION CONSIDERATIONS

5.1. Feature Extraction

The characteristics of the feature space impact the capabilities of the FBDL architecture. Every feature is a descriptor that is used to organize the segments of audio in the delay line. Different features afford different ways to organize the segments of audio, and this organization affects segment selection during the targeting method. Importantly, each feature is a characteristic that can be compared to the target position in order to determine a segment's viability. For example, the presence of a “loudness” feature or a "noiseness" feature in the feature set allows for volume or noise to affect a segment's viability respectively.

5.1.1. Feature Impact

A given feature is only as useful as the amount of variation present among the population of segments in the delay line. In other words, “noisiness” provides no useful differentiation in a population of segments that all have the same amount of noise. The same is true for pitch, loudness, or any other descriptor. With this notion, variations in acoustic properties among different sound types may necessitate the selection of distinct features for optimal signal processing. For example, with percussive sound sources, measurements of energy and noise and change in spectra are likely to be more useful than specific pitch related features such as fundamental frequency, as opposed to tonal sources which are likely to benefit from the opposite. Part of exploring the feature space is discovering which set of features provide the most utility for a given sound source.

5.1.2. Feature Set

Access to a broad feature set is important in order to maximize the customization possibilities during audio segment selection. To this end many features are available, including some that measure similar characteristics through different means. The full feature set is as follows: MFCC, Spectral Centroid, Spectral Bandwidth, Spectral Rolloff, Spectral Flatness, Spectral Flux, Spectral Contrast, Short Time Energy, Short Time Variance, RMS, and Fundamental Frequency Estimation. Feature sets can be streamlined by selecting a more manageable set that maximizes contrast and minimizes redundancy.

5.1.3. Dimension Reduction and Clustering

Dimension Reduction and Clustering algorithms are used in the implementation's GUI as means of simplifying the complex n-dimensional feature space into a more comprehensive, understandable, 2-dimensional representation. Dimension reduction is computed using Principal Component Analysis (PCA) [17] and clustering is computed using DBScan [18]. While versions of these algorithms have often been used in concatenative synthesis applications and other audio database visualizations to reduce dimensionality and present clustering structures, there are unique considerations involved with this architecture due to the ever-changing dynamic database (delay line buffer), compared to past applications with static databases. Namely, this includes stabilization of the analysis in the presence of rapid change, especially for dimension reduction, where small changes may cause the reduction to flip orientation or change basis dramatically. Real-time PCA is still an open problem in the data science community [19].

5.2. Concatenation

Operations required for concatenation synthesis can also materially impact the sonic quality of the delay line. Most importantly, the treatment and combination of delay line segments requires the greatest consideration, where particular focus may be given to segment definition, windowing, number of segments, layering of segments, and segment effects processing.

5.2.1. Segment Definition

Segment definition plays a crucial role in the concatenation synthesis of the delay line audio. Segment delineation can be determined automatically via onset detection, or from designation of regular units of time such as length in samples or beats per minute. The segment size has a substantial effect on the sound quality of the concatenation. Segment size must be set at a minimum such that it can constitute a frame for spectral analysis. For our implementation, the default size is 2048 samples with 50% overlap. Segment size may be tuned for different use cases, where short segments (grains) sound more textural when stitched together, maintaining timbre but not temporal events, and longer segments (syllables) sound more like musical events strung together in sequence.

5.2.2. Windowing and Overlap

Windowing is applied to each segment in order to smooth the transition between disparate audio segments. For our implementation, a Tukey windowing function [20] is used, which has a sinusoidal onset and offset, and a flat band in the center. Control of the center bandwidth affects the quality of the concatenated audio stream, where a larger bandwidth yields greater individual presence of each segment and a smaller bandwidth results in less individual presence as onset / offset periods of segments meld together. Segments read from the delay line may be overlapped during the onset and offset periods of their windowing function in order to more seamlessly stitch them together. This is particularly important with small segment sizes in order to mitigate the introduction of unwanted spectral artifacts. For the special case of configuring the delay line to perform as a traditional delay line, the center bandwidth may be set to the size of the entire window with zero overlap.

5.2.3. Number of Segments

The number of segments determines how many segments are being read from the delay line at any given time. Each additional segment is another tap into the delay line. This parameter greatly increases the textural capabilities of the concatenation process. Additionally, multiple segments may layer to produce chorus-like effects, as similar audio segments are combined together with slight pitch and time offsets. When combining large numbers of segments, they are typically offset from each other for the following reasons. Firstly, this naturally offsets the onset and offset periods of the segment windows. Secondly, this generally allows for more variation in the segments selected, and if multiple copies of the same segment are selected, their constructive amplification is mitigated. Finally, this is computationally advantageous, as it spreads out the targeting queries between more calls to the audio processor.

5.2.4. Adding Effects

A multitude of effects can also be applied to the concatenated segments read from the delay line. These include speed and pitch shifting, reverse playback, panning, waveshaping, and more. Effects can be applied universally across all segments or uniquely to each segment read from the delay line. Variation in how the effects are applied allows for more diversity of texture. For example, this is important for pitch and panning, as it allows for chorus-like pan and pitch width effects, where pan and pitch offsets are applied to segments spread evenly around a center value.

5.2.5. Parameter Mapping

All the effects and parameters can be mapped to the read segments in different ways. They can be manually designated by the user or parametrically automated to produce interesting results. One of the unique affordances of the FBDL architecture is the access to an array of analysis of all the contained audio, which can be used to achieve powerful real time automatic control of parameters. Specifically, effects can be mapped to any of the feature axes of the current target position or the feature set of each individual segment read with custom graphs. This allows for numerous possibilities, including automatic equalization of segment volume, pitch normalization of segments, silencing noisy sounds, etc., as well as many custom behaviors for other specific goals. Generally, the characteristics of the sound can uniquely determine audio effects processing for each segment, allowing for endless customization of the playback of audio from the delay line.

6. USE CASES

The Feature-Based Delay Line architecture promotes many different use cases which were explored in our implementation. As an expansion of the traditional delay line, it fulfills the same functionality. However, with increased control over the delay behavior and extra affordances of the architecture, it expands the boundaries of traditional use cases into new territory for sound design. Furthermore, although currently a high level tool, we believe that future iterations of the FBDL approach may have potential use cases as a lower level signal processing component.

6.1. General Purpose Delay

Typical delay use cases can be enhanced in many ways. Subtle new affordances can be introduced with conservative FBDL settings, such as small target radius, no segment layering, and minimal segment effects processing. For example, targeting parameters can subtly enhance traditional use cases by affecting segment selection, such as expanded target radius introducing variation into the delayed signal. Additionally, unwanted portions can be filtered from the input signal using parameter mapping, via mapping characteristics of unwanted audio such as noisiness to volume. Alternatively, desirable sections of the input signal can be accentuated by linking characteristics of such segments to parameters such as pan, pitch, feedback amount, or target radius, increasing the possibilities of the effect.

6.2. Textural Audio

The introduction of concatenation synthesis into the FBDL promotes a unique use case for textural audio and layering. By increasing the intensity of the FBDL settings, including expanded targeting radius, increased number of concurrently read segments, and more liberal segment effects processing, the delayed signal can be pushed more towards the textural “audio mosaic” realm. With this, the well established traits of prior concatenative synthesis applications are exhibited, while still maintaining the link to the rhythm and quality of the input audio. The texture created either can stand alone, or be layered on top of the dry signal. The unique strength of the dynamic corpus of the FBDL is exhibited here, as the texture melds with the the original source audio in real time, adding additional layers of timbre.

6.3. Resonance and Comb Filtering

The FBDL presents an interesting use case in comb filtering and resonant delay modeling, due to the expression of the architecture with very short delay times. A selective comb filter effect can be created using a small but non-zero target radius, where the filtering will be predominantly active, as the reference segment will be selected at sub-25ms delay time, but sometimes inactive, when a non-reference segment is selected further back in the delay line. Similarly, with high feedback amounts, novel resonant effects can be achieved. The frequency at which the signal is delayed introduces resonant spectral content, which compounds as it repeatedly feeds back into the delay line. Targeting parameters provide a selective and natural way to periodically break out these feedback cycles, allowing for interesting but manageable resonant delay effects. These approaches can be adjusted via targeting parameters and interacts with concatenation parameters and parameter mappings in interesting ways. This also serves as a high-level example of how the FBDL might be used to enhance existing signal processing operations that make use of delay lines.

7. EVALUATION

Evaluation in concatenative synthesis systems has often historically been lacking [12, 21], due to the creative and / or subjective goals of the creator, often in the role of composer. Nonetheless, we maintain that the evaluation of the Feature-Based Delay Line is crucial for the advancement of the delay line as a signal processing component and as an audio effects processor. To this end, a prototype of our FBDL architecture was realized as a JUCE C++ plugin. Specifically our prototype should encompass the possibilities of the traditional delay line, while also introducing new dimensions in the space, expanding the potential range of behaviors. Utilizing this implementation, we conducted experiments to evaluate performance both as a traditional delay line and with new capabilities, such as targeting radius and parameter mapping. These experiments were conducted in isolated circumstances to allow for targeted assessment.

7.1. Delay Line Variation Experiment

As previously stated, the Feature-Based Delay Line is designed to incorporate traditional digital delay functionalities while also introducing new possibilities. The present study aims to evaluate the FBDL’s performance as a basic delay line and to analyze the changes in output as the target radius is introduced. The experiment pursues two main objectives. The first objective is to compare the delayed signal from the FBDL with that of a standard digital delay line using neutral settings and consecutive increases in target radius. The second objective is to assess the impact of the introduction of the target radius on the output signal and its feature analysis data.

To achieve these objectives, measurements were taken by computing the differences in the waveform and feature analysis between the control signal and the FBDL signal with different target radii. The focus was on two specific results: (1) whether the output signal from the FBDL architecture is identical to that of the control delay under neutral settings, and (2) how much the introduction of target radius affects the output signal and its feature analysis data.

Figure 8 presents the results of the experiment. It shows that with a target radius of zero and default parameters, there is no meaningful difference between the delay signal from the FBDL and the control signal. However, as the target radius increases, the differences between the waveform and feature analysis of the two signals also increase. These differences are strongly correlated with the target radius. The plot indicates a contour due to a spike in variation introduced as the target radius encompasses clusters of segments. This is followed by a slight plateau until the target radius expands sufficiently to include different clusters, eventually resulting in complete randomness at a target radius of 1.0. The shape of this contour may vary depending on the distribution of points throughout the feature space, but it will always be positively correlated with the target radius (with some expression of randomness due to nondeterministic segment selection).

These findings demonstrate the FBDL's ability to encompass the behaviors of a typical delay line and evaluate one of the new axes (targeting, specifically target radius) introduced into the possibility space of the architecture, under circumstances that isolate that particular axis.

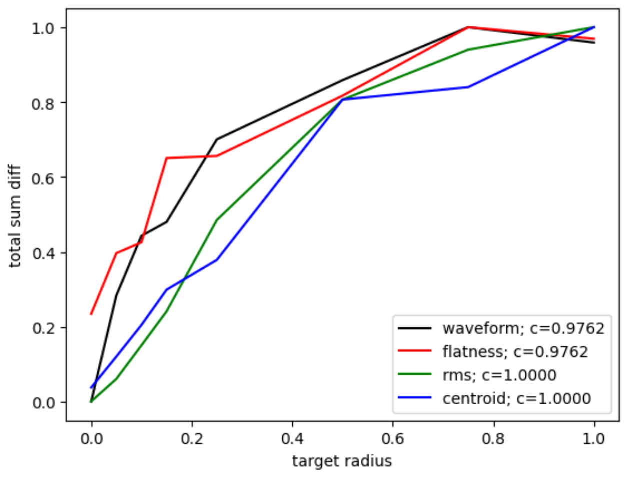

Figure 8: *Waveform & Feature Differences vs. Target Radius.*

7.2. De-Essing Experimental

There are numerous ways to take advantage of the FBDL's utilization of parameter mapping to access analysis of the delay buffer. For our second experiment, de-essing presented itself as an interesting ability of the architecture and an apt candidate for evaluation. De-essing is a well-defined, isolated practice with a clear connection to feature analysis. For example, sibilance in a vocal sample corresponds to increased measures of noise. Specifically, this experiment tests the de-essing capabilities of parameter mapping by using a mapping of spectral flatness to volume and compares the results to a commercial de-essing plugin (set to max strength).

The de-essed signals from both the FBDL and the commercial de-esser are compared to each other and to the control signal through waveform and feature analysis differences. The FBDL's signal is recorded with a delay time equal to the minimum analysis frame size, then time shifted to be in sync with the other waveforms for comparison. The amount of noise in the signals is computed by summing the multiplication of each sample by its spectral flatness value. The remaining noise ratio is the amount of noise in the de-essed signal divided by the amount of noise in the control signal. The removed noise ratio is the amount of noise in the difference between the de-essed signal and the control divided by the amount of noise in the control signal.

As shown in Figure 9, the de-essing created by the parameter binding is effective at filtering out noise from the control signal, removing 45.62%, compared to 23.53% removed by the commercial de-esser. Note that the sum of the remaining and removed noise sums roughly to the amount of noise in the unaffected waveform. The de-essed waveforms are plotted in Figures 10 and 11, with removed signal highlighted in red.

These results show FBDL's success as a de-esser within this scope of evaluation, as well as serving nicely to display the efficacy of parameter mapping as a general technique, which could be applied in other ways towards unique ends.

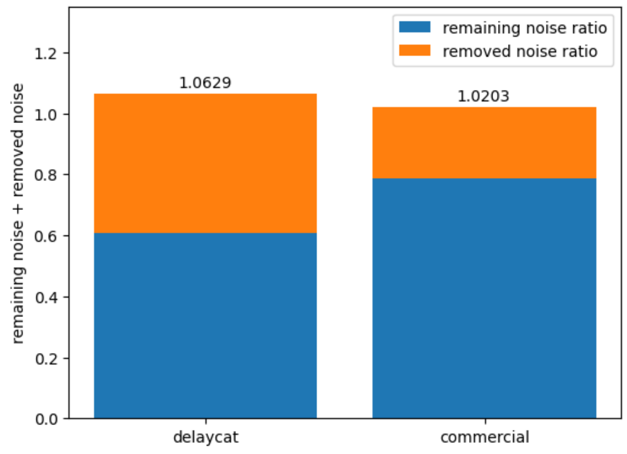

Figure 9: *Removed & Remaining Noise*

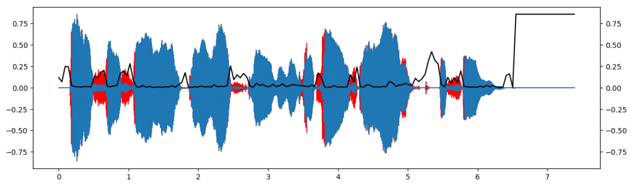

Figure 10: *FBDL De-Essing Effect*

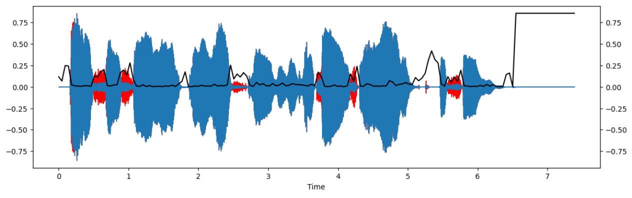

Figure 11: *Commercial De-Essing Effect*

8. CONCLUSION

The union of concatenation synthesis and dynamic delay buffer into an intelligently guided Feature-Based Delay Line (FBDL) offers unique possibilities both as an expansion of the traditional delay line and as an application of concatenation synthesis. With this architecture, we aim to inspire further exploration of music information retrieval and concatenative synthesis in the area of audio effects processing, and innovate upon the prior applications of concatenative synthesis by introducing a real-time dynamic database and expandable targeting method to traverse the feature space based on delay time. Furthermore, expansion on the delay line as a fundamental component may result in progress throughout related areas of signal processing.

Future work on the FBDL architecture will address unique considerations of this approach, such as segment alignment and feature analysis with an arbitrary delay time, along with real-time dimension reduction and clustering integration into the targeting and parameter mapping parts of the architecture as additions or substitutions in the feature set. Additionally, continued development in FFT optimization, especially through GPU accelerated implementations [22] and / or dedicated FFT processing hardware [23] will limit the amount of error in the architecture and expand its potential. Finally, we look to conduct a detailed performance evaluation and seek qualitative user feedback from sound designers to identify areas of improvement in our design and implementation.

We are eager to share our approach with the broader audio and music community. A video demonstration, as well as builds of the plugin, experimental notebooks, and performance evaluation notes are attainable via the project repository located at `https://github.com/NiccoloAbate/DelayCat`.

9. REFERENCES

[1] Diemo Schwarz, “Current research in concatenative sound synthesis,” in *ICMC*, 2005.

[2] Iannis Xenakis and Mrs John Challifour, *Formalized Music: Thought and Mathematics in Composition*, Bloomington: Indiana University Press, 1971.

[3] Andrew J. Hunt and Alan W. Black, “Unit selection in a concatenative speech synthesis system using a large speech database,” *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1, pp. 373–376 vol. 1, 1996.

[4] Joseph P Olive, “Section introduction. concatenative synthesis,” in *Progress in Speech Synthesis*, pp. 261–262. Springer, 1997.

[5] Diemo Schwarz, “Concatenative sound synthesis: The early years,” *Journal of New Music Research*, vol. 35, pp. 3–22, 03 2006.

[6] Ari Lazier and Perry Cook, “Mosievius: Feature driven interactive audio mosaicing,” in *Digital Audio Effects (DAFx)*. Citeseer, 2003.

[7] Diemo Schwarz, Grégory Beller, Bruno Verbrugghe, and Sam Britton, “Real-time corpus-based concatenative synthesis with catart,” in *Proceedings of the 9th International Conference on Digital Audio Effects, DAFx 2006*, 2006.

[8] Grégory Beller, “Gestural control of real time concatenative synthesis,” *ICPhS*, 09 2011.

[9] Michael Zbyszyński, Balandino Di Donato, Federico Ghelli Visi, and Atau Tanaka, “Gesture-timbre space: Multidimensional feature mapping using machine learning and concatenative synthesis,” in *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2019, pp. 600–622.

[10] Steven An, Doug James, and Steve Marschner, “Motion-driven concatenative synthesis of cloth sounds,” *ACM Transactions on Graphics - TOG*, vol. 31, 07 2012.

[11] “Echobit,” `https://echobit.myshopify.com`, Accessed: 2022-05-01.

[12] Cárthach Ó Nuanáin, Perfecto Herrera, and Sergi Jordà, “Rhythmic concatenative synthesis for electronic music: Techniques, implementation, and evaluation,” *Computer Music Journal*, vol. 41, pp. 21–37, 06 2017.

[13] Carl Moore and William Brent, “Interactive real-time concatenative synthesis in virtual reality,” in *Proc. ICAD*, 2019, pp. 317–320.

[14] Eric Tarr, *Algorithmic Reverb Effects*, pp. 349–379, Routledge, 06 2018.

[15] Udo Zölzer, *DAFX: Digital Audio Effects: Second Edition*, John Wiley & Sons, 03 2011.

[16] Roger A. Horn and Charles R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.

[17] Michael Greenacre, Patrick Groenen, Trevor Hastie, Alfonso Iodice D’Enza, Angelos Markos, and Elena Tuzhilina, “Principal component analysis,” *Nature Reviews Methods Primers*, vol. 2, pp. 100, 12 2022.

[18] Martin Ester, Peer Kröger, Joerg Sander, and Xiaowei Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 01 1996, vol. 96, pp. 226–231.

[19] Ranak Roy Chowdhury, Muhammad Abdullah Adnan, and Rajesh Gupta, “Real-time principal component analysis,” *ACM/IMS Transactions on Data Science*, vol. 1, pp. 1–36, 06 2020.

[20] “Tukey window,” `https://www.mathworks.com/help/signal/ref/tukeywin.html`, Accessed: 2022-05-16.

[21] Cárthach Ó Nuanáin, Perfecto Herrera, and Sergi Jordà, “An Evaluation Framework and Case Study for Rhythmic Concatenative Synthesis.,” in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, New York City, United States, Aug. 2016, pp. 67–72, ISMIR.

[22] Dmitrii Tolmachev, “Vkfft - a performant, cross-platform and open-source gpu fft library,” *IEEE Access*, vol. PP, pp. 1–1, 01 2023.

[23] Joseph Ja’Ja and Robert Owens, “An architecture for a vlsi fft processor *,” *Integration, the VLSI Journal*, vol. 1, pp. 305–316, 12 1983.