A General Framework for Visualization of Sound Collections in Musical Interfaces.

applied sciences

MDPI

Article

A General Framework for Visualization of Sound Collections in Musical Interfaces

Gerard Roma [1,*], Anna Xambó [2], Owen Green [1] and Pierre Alexandre Tremblay [1]

[1] Centre for Research into New Music (CeReNeM), University of Huddersfield, Huddersfield HD1 3DH, UK; o.green@hud.ac.uk (O.G.); p.a.tremblay@hud.ac.uk (P.A.T.)
[2] Music, Technology and Innovation (MTI[2]), De Montfort University, Leicester LE1 9BH, UK; anna.xambo@dmu.ac.uk
* Correspondence: g.roma@hud.ac.uk

Abstract: While audio data play an increasingly central role in computer-based music production, interaction with large sound collections in most available music creation and production environments is very often still limited to scrolling long lists of file names. This paper describes a general framework for devising interactive applications based on the content-based visualization of sound collections. The proposed framework allows for a modular combination of different techniques for sound segmentation, analysis, and dimensionality reduction, using the reduced feature space for interactive applications. We analyze several prototypes presented in the literature and describe their limitations. We propose a more general framework that can be used flexibly to devise music creation interfaces. The proposed approach includes several novel contributions with respect to previously used pipelines, such as using unsupervised feature learning, content-based sound icons, and control of the output space layout. We present an implementation of the framework using the SuperCollider computer music language, and three example prototypes demonstrating its use for data-driven music interfaces. Our results demonstrate the potential of unsupervised machine learning and visualization for creative applications in computer music.

Keywords: data-driven music interfaces; dimensionality reduction; music visualization; sound collections; sound visualization; machine learning

Citation: Roma, G.; Xambó, A.; Green, O.; Tremblay, P.A. A General Framework for Visualization of Sound Collections in Musical Interfaces. *Appl. Sci.* **2021**, *11*, 11926. https://doi.org/10.3390/app112411926

Academic Editor: Philippe Esling

Received: 12 November 2021
Accepted: 8 December 2021
Published: 15 December 2021

1. Introduction

Computers, in their many incarnations, are nowadays ubiquitous at different points in most music creation and production workflows. One reason for this prevalence is the convenience of digital storage: compared with analog storage media such as magnetic tape, digital storage makes it much easier to access and manipulate large quantities of audio. Another is the flexibility and easy access to all sorts of techniques for audio synthesis and manipulation.

The audio recordings used in music creation and production can come from a diversity of sources: from crafted loops, to field recordings, performances, commercial music releases, or instrument samples. In many genres and practices, large collections of recordings play an important role from the very early stages of the workflow.

However, most available software and hardware for music production are dominated by traditional paradigms, often established in the analog era or the early days of digital technology. While Digital Audio Workstations (DAW) offer many features beyond analog hardware, their graphical interfaces are, in most cases, still modelled after mixing consoles and tape recorders. Many software samplers feature skeuomorphic user interfaces that emulate with surprising detail the interface of early hardware samplers and sampling synthesizers. Computer music languages such as Max, Pure Data or SuperCollider [1,2], mostly based on the Music *N* paradigm [3] (in turn inspired by analog modular synthesiz-

ers) have also evolved relatively slowly. In these environments, it is still cumbersome to manage large quantities of audio samples.

In recent years, advances in signal processing and machine learning have improved our ability to interact with large quantities of digital information. Most research has focused on supervised methodologies, where an algorithm learns from some known association of digital data to labels. Supervised approaches are, however, of limited use in the early stages of creative processes like music: in these stages, everything is subject to change, and personal interpretations are often more relevant than established conventions. As an example, a musician can easily create a database of recordings using one particular instrument or device. A model pre-trained for conventional sound categories (say, different musical instruments) would not apply to that situation. Contrastingly, unsupervised algorithms such as data clustering or dimensionality reduction could be used to reveal groupings of sounds that are particular to the distribution of audio features in those recordings. While the groupings obtained with these techniques may not directly map to the user's expectations, they can be used to suggest new perspectives and creative possibilities with respect to how the sounds in the recordings may relate to each other.

With respect to computer music applications, research on unsupervised machine learning has mostly focused on interaction with feature spaces. This interaction may be mediated by 2D or 3D visualization, or by different kinds of spatial queries such as gestures in physical space, or audio input. Most often, research has progressed through the development of experimental prototypes [4–8]. Such prototypes have demonstrated the usefulness of unsupervised machine learning for music creation, but are often tied to specific choices in descriptors, spatial mappings, or interaction designs.

In this article, we propose a general framework for devising common pipelines for spatial interaction with sound collections. Our work seeks to enable creative practitioners interested in devising their own interfaces based on unsupervised machine learning, in the same way that open-ended creative coding environments such as Max, Pure Data or SuperCollider have enabled the creative use of audio signal processing techniques beyond packaged solutions. Our research thus seeks to answer the question of how to design a modular framework that supports the development of custom interfaces for using large collections of sounds in music creation workflows.

This article is organized as follows: first, we summarize the state of the art with respect to content-based visualization of sound collections. In Section 3, we describe the proposed framework for the modular design of visualization-based interfaces. We discuss how this generalizes previous work focused on specific prototypes. In Section 4, we evaluate some of the contributions of the framework with four audio datasets. In Section 5, we describe an implementation, in the form of a library for the SuperCollider language, based on the Fluid Corpus Manipulation Toolkit (FCMT) [9]. We present some examples of interfaces developed with this library in Section 6, and discuss our findings and limitations of this work in Section 7. We add some concluding remarks in Section 8.

2. Background

Creative stages of music production are typically driven by musical intuitions and auditory cues. In this context, dealing with labels and file systems can be disruptive, which hinders the use of large collections of sounds. Non-verbal interfaces for interacting with large collections of sounds (from very short segments in the order of a hundred milliseconds to longer audio recordings such as music loops) are valuable, although still rare.

A good deal of research has focused on automatic audio analysis for creating interactive systems based on 2D plots of sound collections. One basic strategy is to use two perceptual sound descriptors (for example pitch and amplitude) as the two axes of the plot. Several works have implemented this idea in interfaces that allow the user to choose the descriptor for each axis [10,11]. These systems suffer from several limitations. First, the direct use of sound descriptors often requires an understanding of concepts related with signal processing and psychoacoustics. Moreover, there is no assurance that a given sound

collection will have an interesting variation along a given set of descriptors. For example, a pitch descriptor may be irrelevant for a collection obtained from drums or environmental sounds. In general, each collection will most likely have its own sonic dimensions beyond a particular choice of descriptors. Second, such descriptors are typically obtained from a frame-level representation, which means they may vary significantly over time. A single value (typically the average over a sequence of frames) is relevant for a very short sound, but it will not be as useful for longer samples.

Automatic descriptors that can capture the diversity in sound collections typically span more than two dimensions. At the same time, different sounds can be related on the basis of non-linear functions of these descriptors. Thus, several works have explored non-linear dimensionality reduction to create automatic low-dimensional maps of sound collections.

An early example of non-linear dimensionality reduction for sound can be found in research on perception of musical instrument timbre [12], where Multi-Dimensional Scaling (MDS) was used to find *timbre spaces*. The same concept has been more recently implemented using Variational Autoencoders (VA) [13]. This concept is limited to the view of timbre as a static quality of pitched sounds and thus not generally applicable to sound collections.

Another early trend was the use of Self-Organizing Maps (SOM) for browsing collections of drum sounds [14] and sound effects [15]. SOMs have the property of naturally producing grid layouts, but they are mostly interpreted as clusters, where multiple data points can be assigned to the same output location.

Most modern non-linear dimensionality reduction approaches are based on neighbour graphs. The IsoMap algorithm [16], based on applying MDS to distances derived from the k-nearest neighbors (KNN) graph, was used in [17]. In [18], the algorithm was extended using the Hungarian algorithm to map the results to grid layouts. More recently, several works have focused on newer dimensionality reduction algorithms. t-distributed Stochastic Network Embedding (t-SNE) [19] uses a probabilistic view of the distances in the original high-dimensionality space to build the neighbourhood graph. This algorithm was used for visualizing sounds from the Freesound database in [5], and for visualising textural sounds in [7]. Uniform Manifold Approximation and Projection (UMAP) [20] is a similar algorithm based on concepts from mathematical topology. The results are similar to t-SNE, but it is significantly faster. It has been used in [6,21].

3. Proposed Framework

In this section, we present the proposed framework for visualization of large sound collections. The framework can be seen as a conceptual abstraction of common data processing pipelines used in music information retrieval (MIR) and general data science that aims at facilitating experimentation by musicians. In previous work [9,22], we have presented basic signal processing and machine learning building blocks for creating arbitrary creative workflows in creative coding environments (CCE). Here, our aim is to provide a higher-level interface that allows intuitive operation by lowering the level of understanding of signal processing and machine learning concepts required for musicians and sound designers, while still retaining a modular interface. Our focus is thus constrained to interaction with spatial layouts of audio collections, with the goal of facilitating the creation of custom musical interfaces. In our implementation, we leverage the abstraction capabilities of the SuperCollider language to provide a high-level programming interface, so that most of the signal processing and machine learning can be controlled through configuration, and user code can be added for the interface.

3.1. Overview

The proposed framework allows devising custom subsets of the pipeline depicted in Figure 1. The core elements encapsulate the functionality for feature extraction (Section 3.3) and pooling (Section 3.4), dimensionality reduction (Section 3.5) and indexing (Section 3.7). Optional steps include audio segmentation (Section 3.2), automatic generation of icons

from audio features (Section 3.3.2), and layout mapping (Section 3.6). Each step can be configured in different ways, depending on the creative application and the nature of the sound collection. We now cover each of the steps in more detail, and conclude with a review of prior work with respect to the dimensions of the proposed framework (Section 3.8).

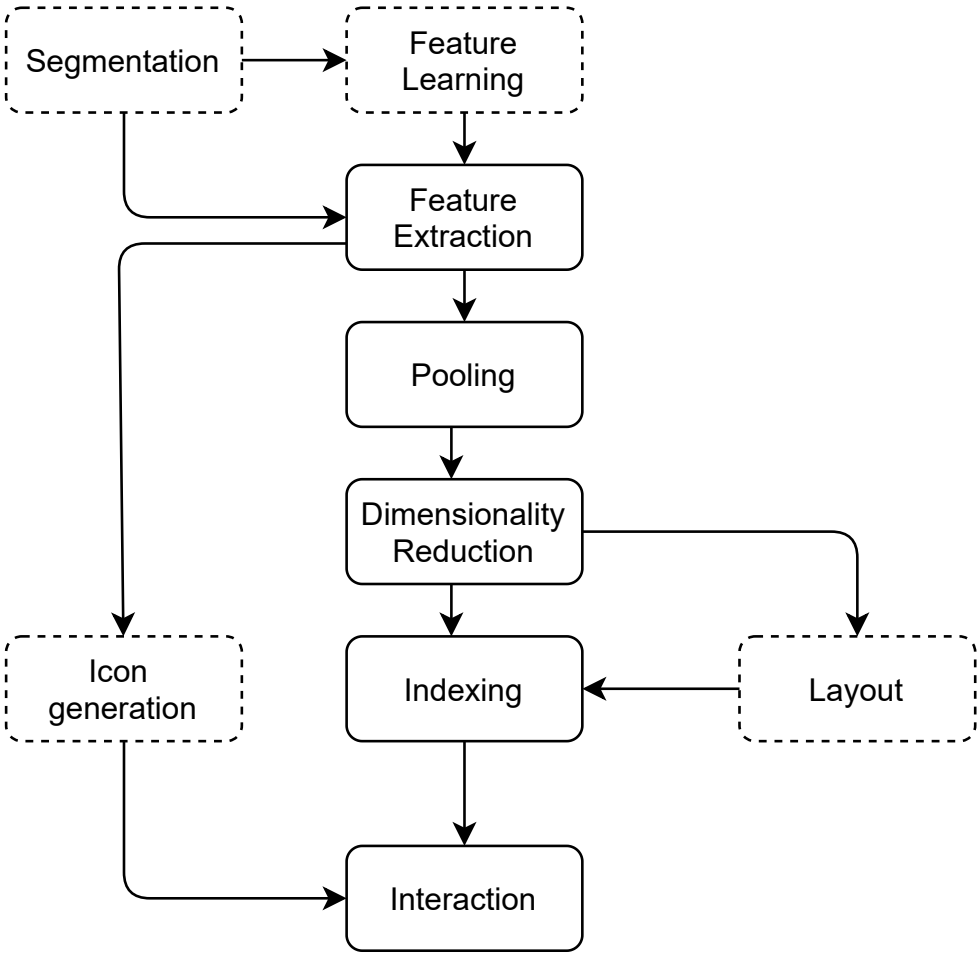


Figure 1. Block diagram of the framework. Solid lines indicate core components, while dashed lines indicate optional components.

3.2. Segmentation

For music creation applications, it is often convenient to work with short segments (e.g., in the order of a few seconds or shorter). From a creative point of view, shorter sounds afford more flexibility for recombination into new sounds or patterns. From a technical point of view, the meaningful description of short time series using feature vectors is also easier, while longer series may contain diverse events which may be more difficult to summarize. Segmentation is thus an important step, unless the source material is already a collection of sounds with a desired length. A common strategy is segmentation by onset detection [23]. Onset detection techniques are very established in MIR and can be used to detect either percussive events or pitch changes, based on different onset detection functions (ODF). Hence they are mostly indicated for musical signals and sounds similarly containing events or short-term spectral changes.

A more versatile strategy is to use Foote's classic novelty algorithm [24]. This algorithm computes a novelty function by convolving a distance matrix obtained from audio descriptors with a two-dimensional checkerboard kernel representing change. Peaks in the novelty function above a user-specified threshold result in segment boundaries. This means that by choosing different input features, Foote's novelty can be used to detect changes in different aspects of the signal (like in the case of different ODFs for onset detection).

However, unlike in the onset detection, the size of the kernel can also be adjusted, which means that changes can be detected at different time scales. In this sense, Foote's novelty can be used as an onset detector but also for segmentation at other time scales.

Our implementation currently focuses on Foote's novelty, with a choice of input features between pitch, chroma features and mel frequency cepstral coefficients (MFCC). The main segmentation parameters are then input feature, kernel size, novelty threshold, and minimum slice length.

3.3. Feature Extraction

The main challenge for the feature extraction step is the potential variety of input signals. In order to facilitate different representations, our framework affords the choice of different features for computing both the location of a sound segment in the 2D space, and the icon of the sound. Since many automatic descriptors have been developed for audio, it is worth highlighting two main considerations with respect to their usefulness in different steps and applications of the framework.

First, some audio features are directly inspired in human perception, which means they may be more easily interpreted by a human listener. Two clear examples are pitch and loudness. The use of interpretable features can be useful when the qualities of the audio material are known and can be captured by these, which can be seen as a "top-down" approach. On the other hand, other feature extraction approaches, such as feature learning, can be used to adapt to the audio material. In this case, the relevant dimensions are not known in advance, but the algorithm is used to learn them from the data, which can be described as a "bottom-up" approach.

Second, an important aspect of audio descriptors is the dimensionality. Multi-dimensional features can be naturally used to create feature spaces, while scalar features are particularly useful for time series visualization. Groupings of scalar features can also be used to create feature spaces, for example, by combining pitch, loudness and spectral centroid.

3.3.1. Position Features

The proposed framework allows visualising collections of sounds in two-dimensional maps. This is based on dimensionality reduction of high-dimensional feature spaces. Different sets of features may be used for top-down or bottom-up interfaces:

- **Feature learning**: automatically learnt features are by definition adaptive. They may be interpretable, but their aim is to capture the main dimensions of the space spanned by the data. We use the system proposed by [25] based on Principal Components Analysis (PCA) and spherical k-means. These features have been applied to audio in different works [26,27]. The underlying feature is the magnitude spectrum processed by a Mel filterbank, so the algorithm can be seen as an adaptive version of MFCCs.
- **Mel Frequency Cepstral Coefficients**: MFCCs, computed as the cepstrum of the output of a Mel filterbank, are a very established representation of the audio spectrum. Their main characteristic is the ability to remove pitch by using less coefficients than Mel bands, so they are particularly useful as a timbre descriptor, but they can also be used as a compact description of the spectrum by using the full range. MFCCs are hard to interpret, with the exception of the first coefficient which represents the energy of the signal. Thus, they can be seen as a midpoint between feature learning and perceptually inspired features.
- **Chroma features**: chroma features are histograms of the energy in the audio spectrum where the bins are defined by notes in a musical scale. As such, they are particularly useful for describing the harmonic aspects of tonal and polyphonic music signals. A number of variants are available in the literature; our implementation is based on the algorithm by Ellis [28].
- **Spectral shape measures**: spectral shape measures are a set of scalar descriptors that measure the distribution of energy in the spectrum. They were popularized by the MPEG-7 standard [29]. Our implementation includes some of the most established

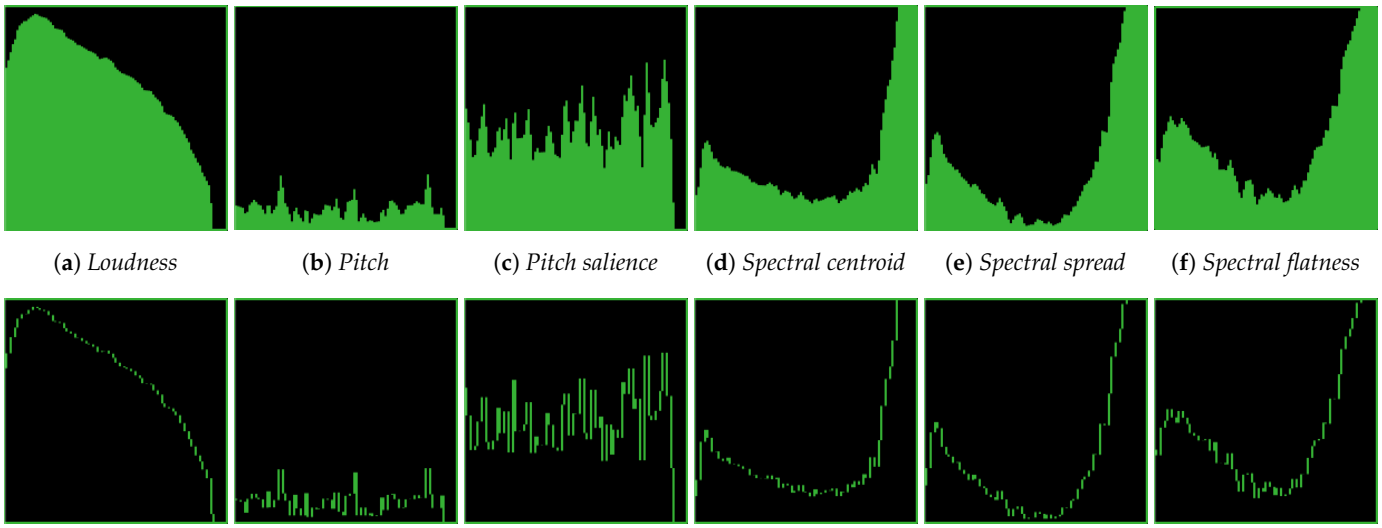
ones: spectral centroid, spread, skewness, kurtosis, rolloff, flatness and crest. As a set, they can be comparable to MFCCs [30], but individually some of them have clear perceptual interpretations. Thus, we use the whole set as a *position feature*, and the most interpretable ones as *icon feature* (see below).

3.3.2. Icon Features

While very short sounds could have negligible variation, in general, sounds projected in the map will contain some perceivable temporal change. Thus, it may be useful to represent some information about the temporal evolution of each sound, in addition to the statistics used to determine the position. In user interfaces, it is common to use graphical icons to represent information about multiple objects. Here, we propose using scalar audio features from each sound to create a graphical representation of the sound's temporal evolution. We call these features *icon features*. Our framework allows choosing among several scalar features. We focus on features with a perceptual interpretation that can be used to provide a visual cue of the contents of the sound. On this basis we propose the following descriptors:

- **Pitch**: pitch is an obvious perceptual feature of musical instruments and human and animal vocalizations. Thus, pitch envelopes can be used as a visual cue. Our implementation is based on the *YINFFT* algorithm [31], which also provides pitch salience measure.
- **Pitch Salience**: pitch salience envelopes can also be useful as visual cues, as many sounds can have identifiable patterns. As an example, the attacks of clearly pitched sounds are often noisy and thus have a low pitch salience.
- **Loudness**: loudness envelopes are perhaps the most obvious cues about audio content and are used routinely for visual representation of audio signals.
- **Spectral centroid**: the spectral centroid gives an indication of the "center of gravity" of energy in the spectrum. It is traditionally related to a perception of brightness [32]. We compute the spectral centroid, as well as the following measures, on a logarithmic frequency scale, so the envelope is more consistent with human perception.
- **Spectral spread**: spectral spread is a measure of the bandwidth of the spectrum, so it typically indicates whether sound is localized in frequency (e.g., monophonic sounds) or occupies a wider frequency range (e.g., noise or polyphonic sounds).
- **Spectral flatness**: spectral flatness measures the flatness of the distribution of energy in the spectrum. Since white noise has a flat spectrum, the envelope gives a cue of the noisiness of the sound over time.

Our framework currently offers a choice between basic shapes and waveform representations. Basic shapes can be either circles or squares, while waveform representations are downsampled versions of the feature time series. In both cases, the average of one icon feature is mapped to the color of the icon. Waveforms may be optionally filled. We found plain waveforms are useful in some cases (e.g., overlaying other interface elements) while filled waveforms work better for smaller icons. Figure 2 shows an example of both types of waveform icons using the different icon features. Note that here the same sound was used for all the examples, and the color corresponds to the average spectral centroid for the whole sound. Since there is only one color per icon, the mapping is only useful in the context of a collection plot.



(**a**) *Loudness* (**b**) *Pitch* (**c**) *Pitch salience* (**d**) *Spectral centroid* (**e**) *Spectral spread* (**f**) *Spectral flatness*

Figure 2. Icon representations of the sound of a dog's bark using different icon features. Each subfigure represents a different shape feature (**a**–**f**). The icon style of the top row uses the *fill* style, whilst the icon style of the bottom row uses the *wave* style.

3.4. Pooling

In order to locate different sounds in a common space, the time series of the position features need to be summarized into a common space. Our system currently uses basic statistics (mean, standard deviation, minimum and maximum), which are applied to both the raw features and their first derivative. For example, for thirteen dimensions in the frame-level feature, this renders a vector of 104 dimensions.

3.5. Dimensionality Reduction

The two-dimensional visualization of sound collections is typically accomplished through the dimensionality reduction of the position features. Many algorithms have been used in the literature for audio collections (see [33] for a review). Here, we focus on three algorithms: Principal Components Analysis (PCA), Multi-Dimensional Scaling (MDS) and Uniform Manifold Approximation and Projection (UMAP).

3.5.1. PCA

PCA is the most popular algorithm for linear dimensionality reduction. Its goal is to find a subspace of the data that preserves as much of the variance in the original data as possible. The dimensions of this subspace are the principal components (PC), and they are found via eigenvalue decomposition of the covariance matrix of the data. PCA can be used as a fast method for visualization by selecting the first two PCs, although it is limited to linear projections of the data. Since the PCs are ranked by importance, it can also be used as a pre-processing step to reduce some noise and redundancy in the dimensions obtained by feature extraction.

3.5.2. MDS

MDS introduces the goal of preserving a distance metric between the points in the data. One of the most classic algorithms for MDS performs an eigenvalue decomposition of a matrix derived from a distance matrix. Thus, it can be used to experiment with different distances, although due to the computational cost of computing the distance matrix, it is not suited for very large collections. The algorithm could also be applied to distance matrices generated from user input.

3.5.3. UMAP

Finally, UMAP is one of the most established algorithms for non-linear dimensionality reduction. Like many modern algorithms for this problem, it can be seen as an optimization

of a system of attractive and repulsive forces computed over a nearest-neighbors graph. In UMAP, the graph is weighted following a probabilistic interpretation with roots in mathematical topology, which results in very good quality visualizations. The algorithm is also significantly faster than other popular algorithms such as t-SNE.

3.6. Layout

One particular affordance of dimensionality reduction algorithms is how they can help reveal clusters of similar points in 2D space. At the same time, this may make it difficult to display sound icons without overlaps, as the points may be arbitrarily close together. One solution to this problem is to map the result of the dimensionality reduction algorithm to a regular grid of points. The mapping of the points from the reduced space to the points in the grid can be seen as an assignment problem and solved using the Hungarian algorithm. In [18], this was applied to the output of the IsoMap algorithm, and an application to audio samples was demonstrated among other examples.

Following this idea, we propose the use of oversampled grids (similarly to [34]) for mapping the output of dimensionality reduction. If the generated grid has many more points than the number of points in the plot (here, the sounds), it will be much easier for the assignment algorithm to find a mapping that respects the original shapes in the plot. On the other hand, if the grid has approximately the same number of points as the original plot, the shapes formed by the points will be distorted, but the available space will be uniformly covered by the points. Our algorithm generates a rectangle spanning the limits of the plot generated by dimensionality reduction. We then generate a grid with at least the same number of points as the original (depending on the oversampling factor). Finally, we use the Jonker–Volgenant variant of the Hungarian algorithm [35] to assign each point to a point in the grid. This allows us to obtain a predictable minimum distance between points, which allows drawing icons, as opposed to colored dots. At the same time, depending on the oversampling factor, we can interpolate between the shapes resulting from the dimensionality reduction step, and a more regular grid that can be used to optimize the available space. Figure 3 illustrates the process using synthetic data (random RGB colors as features) and the UMAP algorithm. The oversampling factor controls the balance between a regular grid and a more clustered plot. At 5× oversampling, the result is very close to the original.

3.7. Indexing

A final step in the data processing pipeline is indexing the data for interactive applications. We normalize the 2D plot and fit a KD-tree algorithm [36] to facilitate nearest-neighbors queries. In this step, we also compute the global statistics for normalization of icon features.

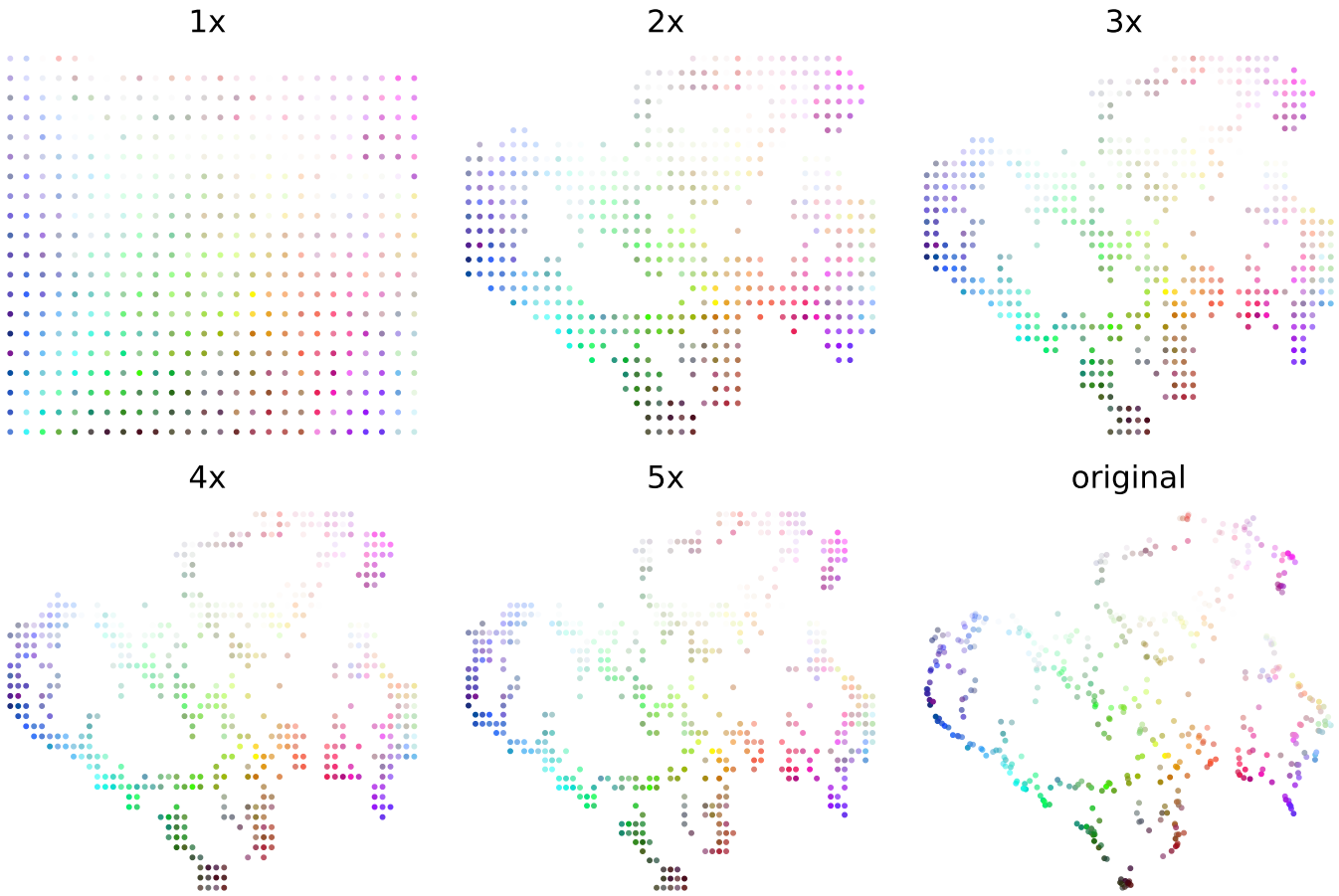


Figure 3. Synthetic example of the grid layout algorithm showing values of oversampling from 1 to 5.

3.8. Relation to Prior Work

As described in earlier sections, our aim is to generalize previous work on the visualization of sound collections. In this section, we analyze previous work in relation to the dimensions of the framework presented in the previous section. Our analysis is presented in Table 1. Our focus is on the potential of visualization for creative applications. For this reason, we have selected works where a working prototype that can be used in this context is presented. We describe 10 relevant systems ranging from 2007 until present:

1. Mused, an interactive scatter plot system for browsing a large database of sounds by Coleman [10].
2. SoundTorch, a virtual flashlight system for browsing large collections of audio by Heise et al. [15].
3. A study on interactive navigation-based search in large sound databases by Schwarz et al. [37].
4. A visualization approach to represent the perceptual qualities of large collections of textural sounds based on tile maps by Grill and Flexer [7].
5. AudioQuilt, a system that flexibly arranges audio clips on a layout based on the user taste by Fried et al. [38].
6. AudioMetro, a system that organizes sounds on an evenly distributed grid by Frisson et al. [39].
7. Floop, a system based on rhythm and timbre analysis for exploring loops in the Freesound database by Roma and Serra [8].
8. Freesound Explorer, a visual interface system for exploring sounds from Freesound based on their timbral properties by Font and Bandiera [5].

9. A study on visualization methods based on perceptual representations of timbre suitable for large libraries of sound samples by Richan and Ruat [21].
10. AudioStellar, a system that enables the generation of interactive visualizations based on large collections of sounds by Garber et al. [6].

Table 1. Comparison of research prototypes for visualization of sound collections.

Authors	Segmentation	Audio Descriptor/s	Reduction Algorithm	Output Layout	Icon Style	Icon Feature
Coleman (2007) [10]	Onset	Spectral centroid Temporal centroid	–	–	–	–
Heise et al. (2008) [15]	–	MFCCs	SOM	–	–	–
Schwarz et al. (2009) [37]	Onset	MPEG-7	Hybrid MDS	–	Color	Descriptor
Grill and Flexer (2012) [7]	–	Several scalar	t-SNE	–	Texture	Custom
Fried et al. (2014) [38]	–	MFCCs Temporal centroid	–	Metric learning + Kernelized sorting	Color	Isomap/Clustering
Frisson et al. (2014) [39]	–	MFCCs Spectral flatness	PCA+t-SNE	Proximity grid	Color Shape	Perceptual sharpness
Roma and Serra (2015) [8]	–	Beat spectrum MFCCs	Force directed layout	–	Spectrogram	–
Font and Bandiera (2017) [5]	–	Chroma MFCCs	t-SNE	–	Color	Tristimulus
Richan and Ruat (2020) [21]	–	Amplitude envelope Roughness envelope Spectral envelope	PCA+UMAP	–	Color Shape Texture	Position Amplitude envelope Supervised learning
Garber et al. (2021) [6]	–	Chroma FFT MFCCs RMS Spectral centroid	PCA t-SNE UMAP	–	Color	Clustering

It can be seen from Table 1 that some features are notably rare. For example, most systems do not include segmentation as part of the pipeline. However, in our view, adding segmentation opens up the possibilities for using longer recordings. While segmentation can be carried out with other tools, integrating it into the same framework allows interactive adjustment of the parameters while seeing the resulting visualization. Thus, we consider the lack of segmentation an important limiting factor of most prior work.

Controlling the layout is also relatively rare. This is necessary to avoid overlaps when icons are used. In [7], the sounds are mapped to a tiled grid, but it is unclear if an automatic algorithm was used, so we would assume the mapping was carried out manually. Fried et al. [38] specifically addressed this issue, but using a significantly expensive method (e.g., they mention the process taking 30 min for 176 samples). They later applied IsoMatch [18] to the same problem, which presumably is faster. We use the same approach as IsoMatch although with different dimensionality reduction algorithms. In [39], an algorithm named Proximity Grid is used [40].

Although the work in [21] is more focused on the evaluation than on the prototype, we included it here because they assess the use of icons, which again is rare among the compared systems. However, their prototype was mostly focused on this experiment and included only 30 samples, so they do not address the issue of overlaps. In addition to this work, only [7,39] feature visualizations for the sound beyond color . The system by the first author in [8] used spectrogram images from the Freesound database, but this approach would be difficult to scale to many sounds due to the complexity of the images. The system in [39] added a unique capability of synthesizing image texture using supervised learning. Our system focuses on waveforms derived from descriptors, which is similar to their mapping to shape, but we allow using different descriptors. The shapes used in [7] were again based on a manual approach.

Finally, an important feature of our framework is the ability to choose between different sets of features for the high-dimensional space that is later projected to the visualization. Most systems use fixed sets of descriptors. Notably, Ref. [6] allows several options, includ-

ing MFCCs and Chroma. It is also worth noting that AudioStellar [6] and CataRT [37] are available to musicians, whereas other systems can be mostly considered research prototypes. However, like [10], the current public version of CataRT uses scalar features as axes of the 2D visualization.

4. Experiments

Beyond the modularity and generality, the proposed framework introduces several novel elements with respect to previous work. In this section, we assess their usefulness through several experiments. First, we assess unsupervised feature learning (Section 4.2), which can be useful for obtaining the feature space without introducing signal processing concepts in user interfaces. We compare the spherical k-means algorithm to traditional audio features. Second, we compare dimensionality reduction algorithms (Section 4.3), with a focus on UMAP. This relatively recent algorithm has been used in some previous works, but has not been evaluated with audio datasets. Finally, we analyze the effect of the grid layout with different oversampling factors (Section 4.4). Like in the case of UMAP, to the best of our knowledge this analysis has not been reported with audio data. Some aspects of the proposed framework, such as the usefulness of icon features, would need to be asessed via user studies, which we leave as future work.

4.1. Methodology

We now describe the methodology using four datasets and three metrics.

4.1.1. Datasets

We used four datasets of audio samples:

- **Drums**: a collection of 750 drum samples collected from different commercial samplers, with labels for 5 instrument classes: *clap*, *hi-hat*, *kick drum*, *ride* and *snare*.
- **SOL**: a subset of 1500 sounds from the OrchideaSOL database used for computer-aided orchestration [41], with labels for main instrument families: *brass*, *keyboards*, *plucked strings*, *strings*, *winds*.
- **Urban sounds**: a subset of 804 sounds from the Urban sounds dataset [42] focusing on 4 classes: *car horn*, *dog bark*, *drilling* and *gun shot*.
- **Gaver**: a collection of 1579 sounds (previously used in [43]) labelled according to Gaver's ecological acoustics taxonomy [44]: *impact*, *rolling*, *scraping*, *deformation*, *drip*, *pour*, *ripple*, *splash*, *woosh*, *explosion*, *wind*.

4.1.2. Metrics

Following [4,45] we used three metrics to assess the plots obtained from dimensionality reduction: Trustworthiness, Continuity and KNN classification error.

- **Trustworthiness**: this metric measures the rank of points in a neighborhood of a given point in the reduced space which are not really neighbors in the feature space:

$$T(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^{N} \sum_{j \in U_i^{(k)}} (r(i,j) - k), \quad (1)$$

where k is the number of points in the neighborhood, N is the number of points, and $r(i,j)$ is the rank of point j with respect to i according to their distance in the 2D space. $U_i^{(k)}$ indicates the set of points that are among the k-nearest neighbors of i in the 2D plot but not in the original feature space. Trustworthiness can be related to the idea of precision in information retrieval evaluation. If the plot is *trustworthy*, sounds that were originally distant should not appear close in the 2D plot.

- **Continuity**: this metric measures the complementary case, which can be related to recall in information retrieval. Points that are originally close should also be close in the 2D plot:

$$C(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^{N} \sum_{j \in V_i^{(k)}} (\hat{r}(i,j) - k), \quad (2)$$

  where $\hat{r}(i,j)$ is the rank of point j with respect to i according to their distance in the original feature space, and $V_i^{(k)}$ indicates the set of points that are among the k-nearest neighbors of i in the feature space but not in the 2D plot.
- **KNN classification error**: the performance of a K-Nearest Neighbors (KNN) classifier in the 2D space (according to the labels of each dataset) also gives an indication of the quality of the plot. Sounds of the same class should appear close in the plot and thus a KNN classifier should be able to generalize from some training samples. We measured the accuracy of a 2-nearest neighbors classifier when varying the fraction of points used for training. In all cases, we used a random partition to label the data, and averaged the results over 10 runs.

4.2. Feature Learning

4.2.1. Settings

We compared MFCCs to feature learning with the spherical k-means algorithm. In this case, we only measured KNN classification error, since the feature spaces are different, which would affect the comparison of continuity and trustworthiness. In both cases, we used windows of 1024 samples (∼20 ms) with hops of 512 samples (∼10 ms). In each case, we selected the best performing parameters when relevant. MFCCs were computed using 40 bands and 13 coefficients. For the spherical k-means features, we used 40 bands and 1024 clusters. We concatenated every consecutive pair of frames into a single frame (a technique often known as *shingling*). Larger numbers significantly impacted performance but would not improve results (a value of 2 did provide an improvement). With respect to pooling, the best results for MFCCs were obtained using mean, standard deviation, min and max for the original time series and first derivative. For spherical k-means features, we used only mean and standard deviation of the raw features.

4.2.2. Results

Figure 4 shows the KNN classification error using MFCC vs. feature learning in each of the dataset. The feature learning algorithm works better for environmental sounds (Urban and Gaver), while MFCCs work better for classical instrument sounds (SOL). The latter sounds are characterized by stable configurations of partials, which is well suited for the classic MFCCs with 13 coefficients. The former sounds are generally less consistent and noisy. Interestingly, there is little difference for the Drums dataset, which has a mixture of noisy and tonal sounds. The environmental sounds datasets generally contain more variety. Thus, it can be expected that feature learning would be a good strategy for noisier and less controlled datasets, while MFCCs would still be better for traditional instrument sounds.

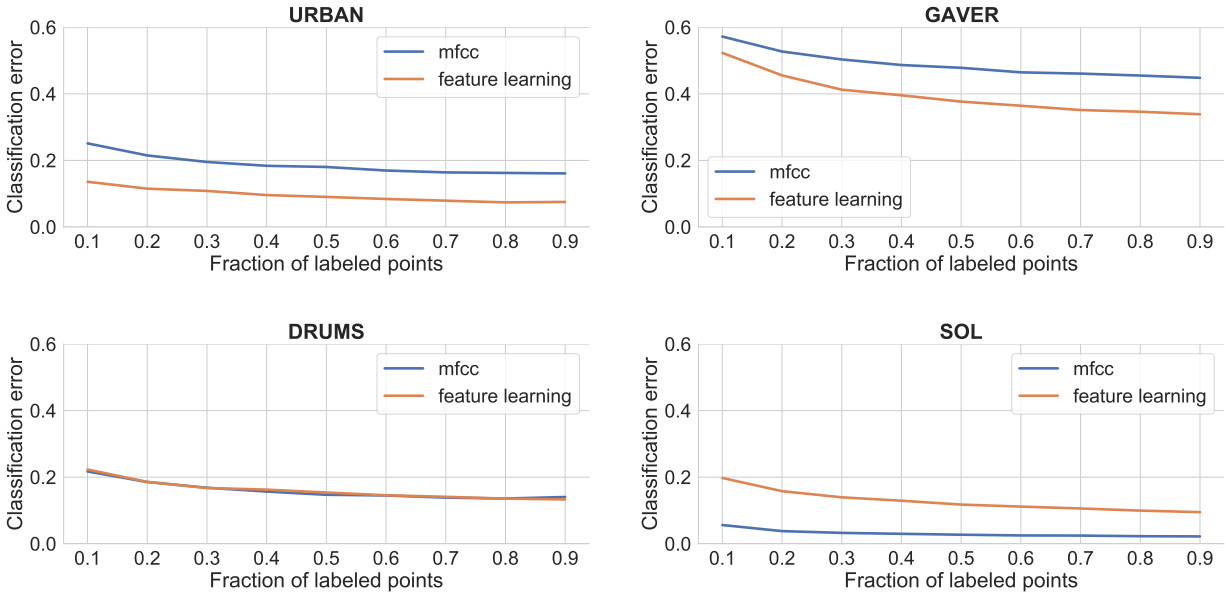


Figure 4. KNN results for different datasets. From top-left to bottom-right: Urban, Gaver, Drums and SOL.

4.3. Dimensionality Reduction

4.3.1. Settings

We compared several dimensionality reduction algorithms with respect to KNN classification error, trustworthiness and continuity. The algorithms are PCA, MDS, Isomap, t-SNE and UMAP. We used the implementations available in the scikit-learn library [46] and the "official" distribution for UMAP [47]. We set the number of neighbors of the UMAP algorithm to 10. For the rest of algorithms, we used default parameters ($perplexity = 30$ for t-SNE). In all cases, we used MFCCs as features.

4.3.2. Results

The results for the comparison of dimensionality reduction algorithms is shown in Figures 5–7. It can be seen that t-SNE and UMAP have generally similar performance and work better than the rest. The only exception is continuity, where all four algorithms score similarly high. This shows that t-SNE and UMAP produce more clustered plots, which include only relevant neighbors for each point (trustworthiness), whereas all four algorithms tend to preserve all neighbors in the feature space as neighbors in the reduced space (continuity). In terms of speed, our experience with the Python/NumPy implementations matches the reported improvement of UMAP over t-SNE [48]. Thus, we would consider UMAP to be better suited for music creation applications.

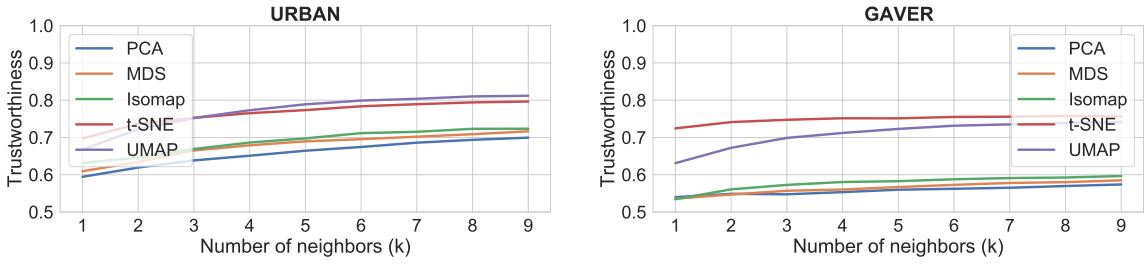


Figure 5. *Cont.*

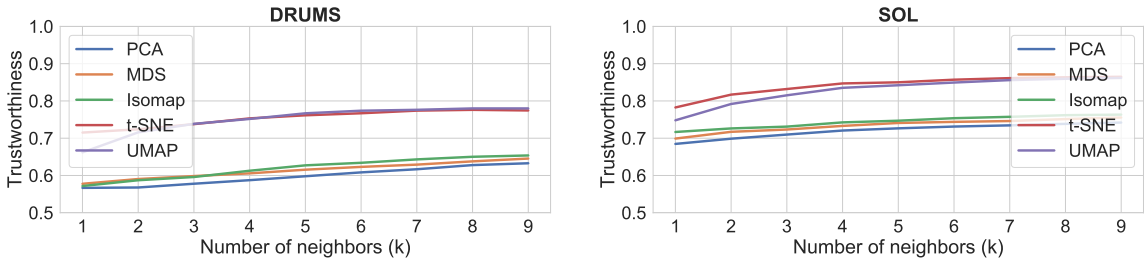


Figure 5. Trustworthiness results with different dimensionality reduction algorithm for each datasets. From top-left to bottom-right: Urban, Gaver, Drums and SOL.

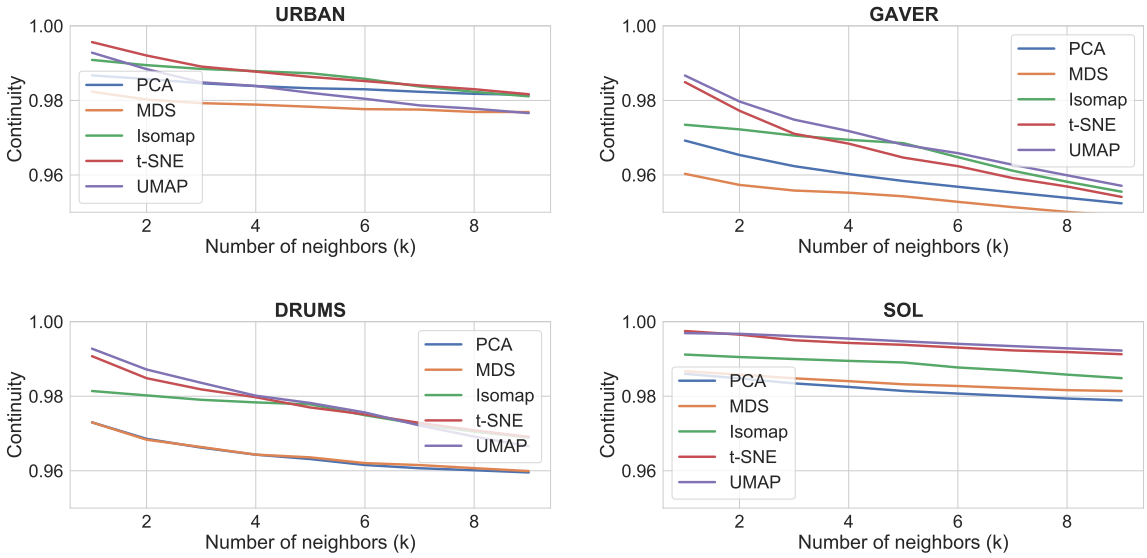


Figure 6. Continuity results with different dimensionality reduction algorithm for each datasets. From top-left to bottom-right: Urban, Gaver, Drums and SOL.

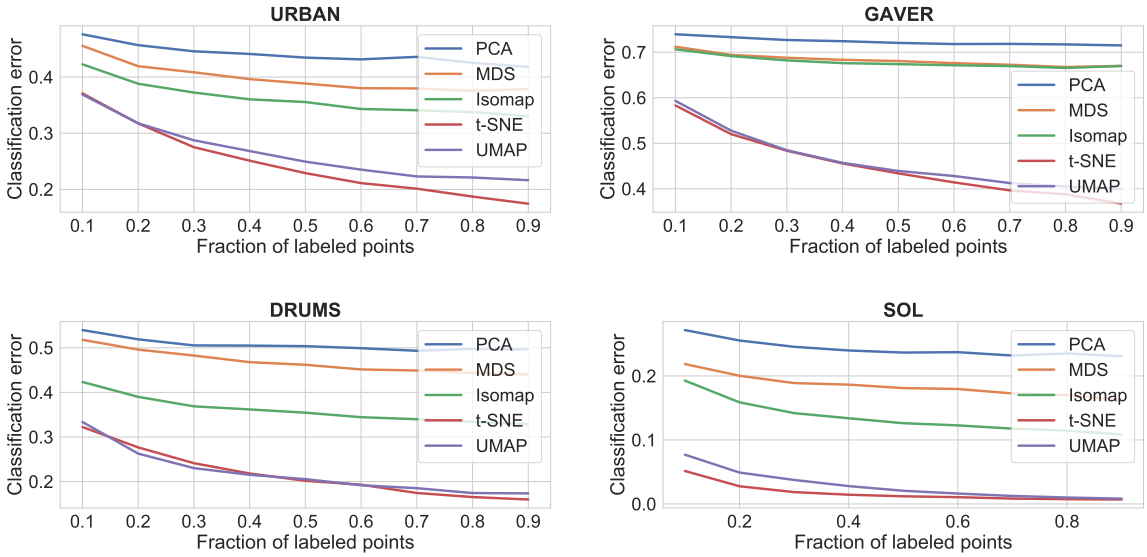


Figure 7. KNN classification error results with different dimensionality reduction algorithm for each datasets. From top-left to bottom-right: Urban, Gaver, Drums and SOL.

4.4. Effect of Layout

4.4.1. Settings

We analyzed the effect of the grid layout and oversampling on trustworthiness, continuity and classification rate in a final experiment. As seen in the previous experiment, these metrics are relatively stable with respect to the fraction of labelled data (classification error) or number of neighbors (trustworthiness and continuity). For this experiment, we fixed these parameters and analyzed each measure when the data are projected to a grid layout with a varying oversampling factor. The grid layout was applied to the output of the UMAP algorithm with MFCC features.

4.4.2. Results

The results were very similar for all four datasets. Figure 8 shows the results for the Drums dataset. The value for each measure for the output of dimensionality reduction (before the grid assignment) is shown in a dashed black line. It can be seen that the three metrics are already close to that value with no oversampling, and approach it quickly when oversampling is used. The continuity measure is already very close with only 2× oversampling. The classification rate benefits from higher oversampling factors, and so does trustworthiness.

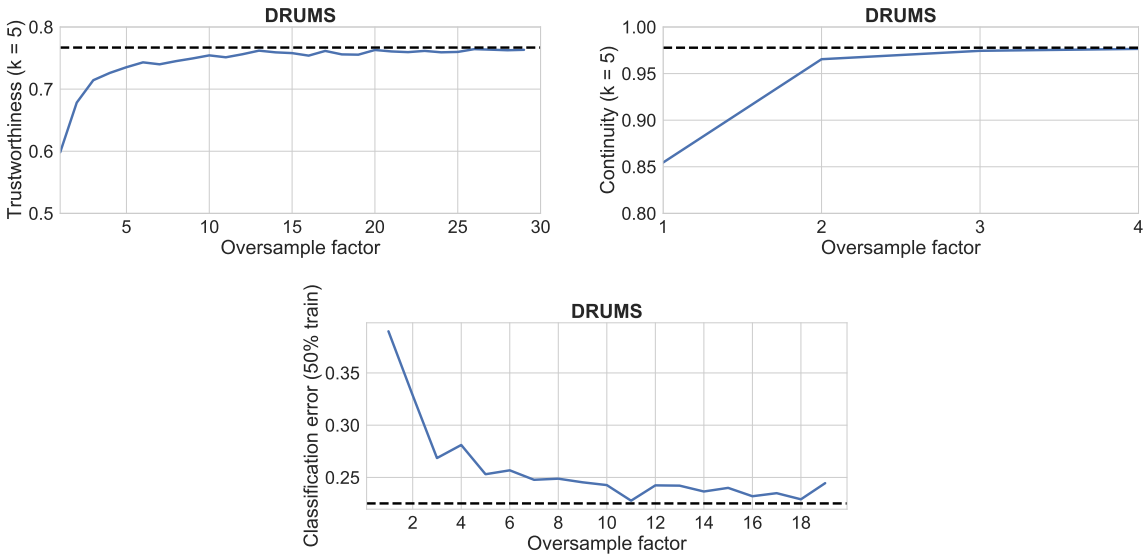


Figure 8. Results for the three metrics when varying the oversampling factor with the Drums dataset. From top-left to bottom-right: trustworthiness, continuity, and classification error.

As seen in Figure 3, the distortion due to the grid assignment draws points that were more distant in the reduced space closer together. This means that we can no longer completely trust that near points will be similar, but at the same time, the grid layout has interesting applications for graphical interfaces. The results show that oversampling the grid is a useful compromise when high precision is desired, at the expense of the optimal usage of the available space, while still allowing for non-overlaping shapes and icons.

5. Implementation

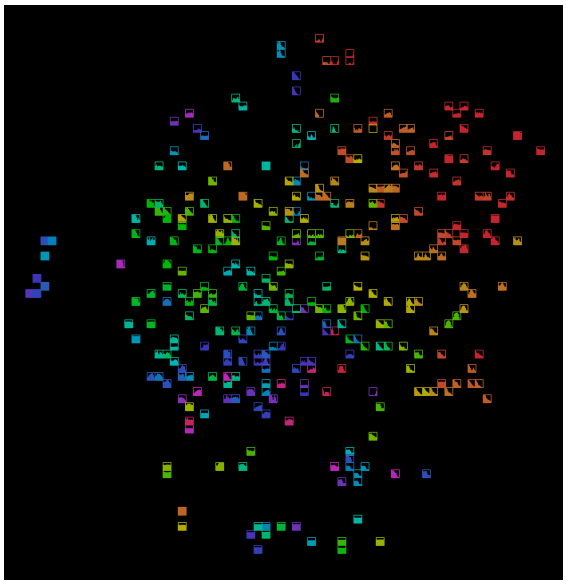
The framework described in the earlier sections was designed with the aim to enable custom interfaces and instruments based on large sound collections. In order to test this in practice, we implemented the framework in *Fluid Corpus Map*, a SuperCollider library on top of FCMT.

Self-made instruments and interfaces are common in computer music thanks to the different available CCEs and languages. However, dealing with large quantities of information and machine learning algorithms can be difficult in this context. As a class-

based object-oriented language, SuperCollider makes it easy to create abstractions that address these difficulties while retaining the modularity. At the same time, it offers a wide range of tools for playing and manipulating audio buffers in combination with different synthesis techniques, as well as creating custom interfaces and graphics.

We designed the library to be configurable for all the options described in the framework. All the settings are in one file, so they can be modified globally in the code, or they can be set at run-time for a given project. This includes segmentation parameters, selection of position and icon features, general analysis parameters (short-time Fourier Transform parameters, number of dimensions and derivatives), icon shape, choice of dimensionality reduction algorithm (as well as number of output dimensions and number of neighbors for UMAP), using the grid and oversample factor, and optionally choosing a specific number of rows or columns for the grid.

One important issue with respect to SuperCollider is that, since the objects in FCMT are implemented as plugins (which run in a separate process in SuperCollider), all interactions with data are asynchronous. This can lead to very complex code. Fluid Corpus Map simplifies this by implementing a job queue and a method chaining interface inspired by the D3 visualization library [49]. Figure 9 shows an example code listing and the resulting plot using the Gaver dataset. The code for the Fluid Corpus Map library is available from https://github.com/flucoma/FluidCorpusMap2 (accessed on 1 November 2021).



```
(
var win = Window.new("", Rect(0, 0, 800, 800));
var fcm = FCM.new();
var plot;

fcm.settings.analysis.positionFtr = \mfcc;
fcm.settings.analysis.shapeFtr = \loudness;
fcm.settings.analysis.colorFtr = \spectral_centroid;
fcm.settings.reduction.useGrid = true;
fcm.settings.reduction.gridSample = 4;
fcm.settings.display.iconSize = 10;
fcm.settings.display.iconStyle = \fill;

fcm.addFolder("gaver_events/")
.makeIndex()
.run{
        plot = FCMPlotView.new(win, win.bounds, fcm);
        win.front;
})
```

Figure 9. An example script using the library, and the resulting plot.

6. Examples

In the previous sections, we presented a modular framework for combining segmentation algorithms, audio descriptors and dimensionality reduction algorithms to create visualizations of sound collections. We have shown an implementation as a SuperCollider library that allows producing such visualization in a few lines of code. This opens up many possibilities for novel interfaces, particularly by making use of other features available in the SuperCollider environment (in prior work [33], we presented a similar set of applications using a combination of Python, SuperCollider and web interfaces. The current version of the library runs purely on the SuperCollider language and server, leveraging our implementation of the described algorithms in FCMT [9]).

We now describe three example interfaces for musical interaction with sound collections using the library. Demonstration videos for each example can be found in the companion website of this article: https://www.flucoma.org/APPLSCI/ (accessed on 1 November 2021).

6.1. Grids of Slices

The first example allows mapping slices to colored squares in a 8×8 grid. Grids of buttons with RGB leds of this size are currently widely used in computer-based music creation. This example makes it easy to map portions of a longer recording (such as, for example, drum patterns or music fragments containing multiple sound events) to squares in the grid, using both position and color as cues of the content of the slices. Two additional sliders allow adjusting segmentation threshold and minimum slice length. The 64 longest slices are selected. A screenshot is shown in Figure 10, using chroma as position features and pitch as color feature, for a rhythmic chord progression. When using this prototype we found that, thanks to the combination of the grid layout, the color mapping, and the possibility of adjusting the segmentation parameters, we could very quickly obtain intuitively playable surfaces out of a wide variety of sound materials.

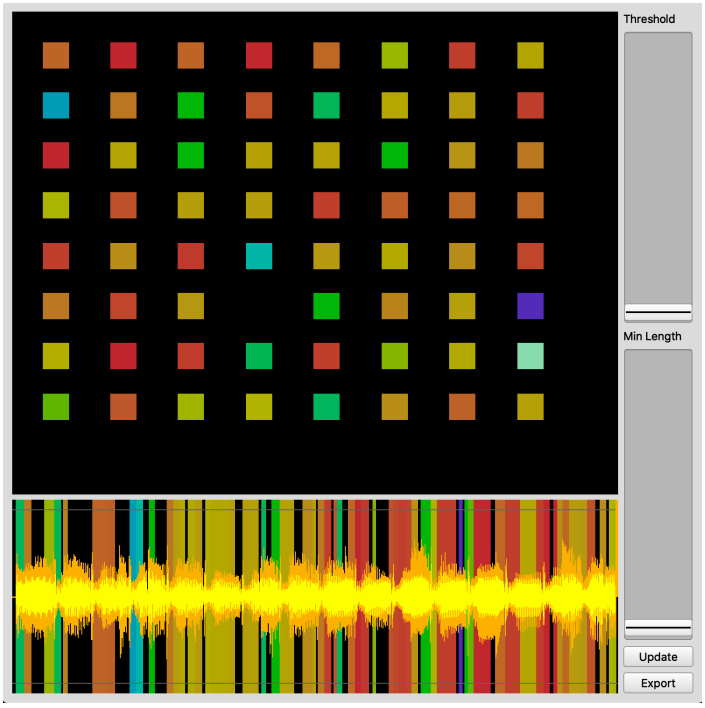


Figure 10. Example of the slicer interface.

6.2. Data-Driven Multislider

The second example was ported to SuperCollider from the one presented previously in [33]. We experimented using dimensionality reduction to create a four-dimensional space instead of a 2D plot, and mapped each of the resulting axes to a slider. Thus, in this case we implemented some graphics code beyond the 2D plots provided by the framework. For each slider we represent the histogram of points along the corresponding dimension using color. We used this interface to create a drone instrument, using a granular synthesizer and a dataset of 500 sounds labelled with the same note from the SOL database. A screenshot is shown in Figure 11. As noted in [33], the four dimensions are not very intuitive, but the prototype works as a creative search interface, and it is easy to memorize and transition between different configurations.

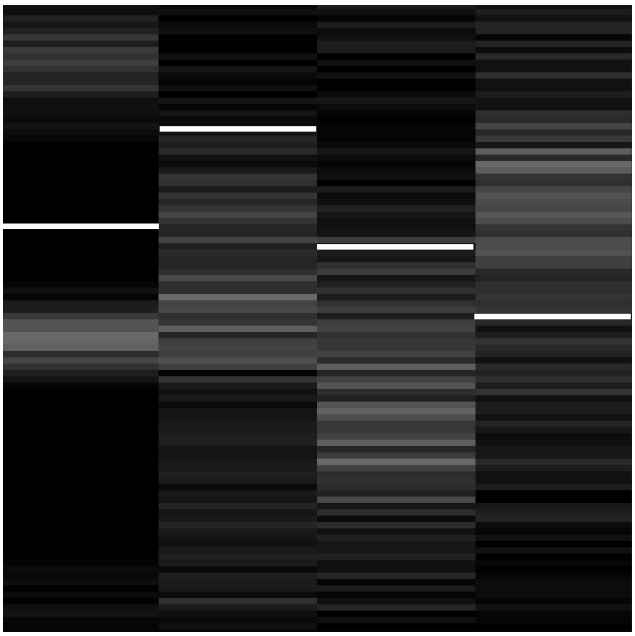


Figure 11. Example of the multislider interface.

6.3. Playing Trajectories

Our last example focuses on playing trajectories in the reduced descriptor space. Here, an oversampled grid layout was used along with waveform-based icons to display the sound collection. MFCCs were used as position features. Loudness was used for the icons, and the spectral centroid was used for color. The interface allows recording and triggering trajectories in the 2D plot. A recorded trajectory is made of a sequence of points in the 2D space. An associated player follows the trajectory and plays the closest sound to each point. A screenshot is shown in Figure 12. Recording envelopes of different parameters is a common feature of digital synthesizers. We found that applying this idea to visualizations of large sound collections allows to easily create phrases that combine movements of timbre with complex temporal dynamics. This capability of triggering sonic trajectories from discrete controls could also be useful to create interfaces for motion-impaired users.

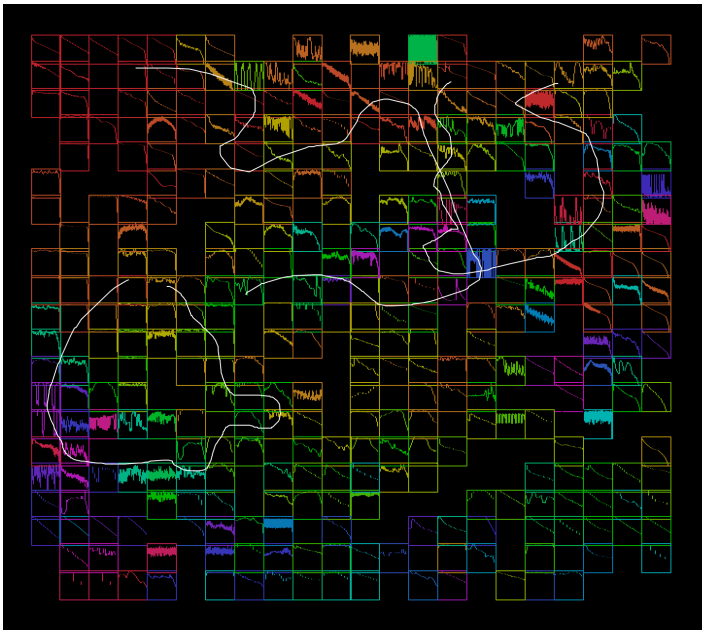


Figure 12. Example of the trajectories interface with three recorded active trajectories.

7. Discussion

In the previous sections, we proposed a modular approach to the dimensionality reduction pipeline as applied to interfaces for interacting with sound collection. We have presented a framework that generalizes previous approaches in terms of features and potential for combinations. In comparison with previous approaches using scatterplots of two scalar descriptors, dimensionality reduction can be used to automate the selection of the axes, freeing users (often musicians and sound designers) from having to decide on the basis of signal processing concepts represented by the descriptors. By introducing a configuration option for position features, we introduce this choice again, although in this context the decision would be mainly to focus either on tonality (using chroma) or timbre (using MFCCs or MPEG7 features), as opposed to picking scalar descriptors. We have introduced the idea of feature learning, which again can be used to automate the choice of features.

In our experiments, we have found that the feature learning algorithm based on spherical k-means can adapt better to sounds with more noise and variability than instrument sounds, such as environmental sounds. However, we have not investigated whether it could be used for tonal material. We hope that future work will contribute towards further automation of the analysis process, with the aim of obtaining intuitive parameters and choices from data.

We have also introduced more flexibility for mapping descriptors to color. Prior work, such as [6], has used data clustering on the 2D plot to provide colors. This could be seen as redundant, since the clusters are already visible in the plot without colors. However, in our experience, it is still important that the color is related to the spatial distribution. While a formal evaluation would be needed to characterize this problem, our experiences so far indicate that it is hard to make sense of plots where color is completely unrelated to position. Again this presents another promising direction for automating the feature selection and extraction process.

We have proposed and evaluated the introduction of grid layouts through the Hungarian algorithm. We have shown that this has two separate advantages. On the one hand, it allows introducing more information about each sound through icons, as it avoids overlaps. On the other hand, the grid can be used to provide more compact layouts that make a more efficient use of screen real estate. As we have shown, oversampling the grid can be used to recover the clusterings of the plots obtained via dimensionality reduction, as reflected in the trustworthiness and classification error measures, while preserving the regularity of the layout that allows representing icons.

With respect to the icons, our approach is deliberately simple. Recent efforts to assess the usefulness of sound icons in visualizations [21] have been inconclusive. In our experience, using simple waveform representations of well-known descriptors is encouraging. In this sense, the creation of meaningful icons from audio features is an interesting design problem by itself that deserves further research. It is worth mentioning that, in early work, we experimented with mapping color to the time series of audio features in the sound icons (as is done for example in the Freesound database). However as mentioned above, we found that when color is decoupled from position, the plot can be hard to interpret, so the combination of two scalar features for the waveform (derived from the time series) and a solid color (derived from the statistic of the time series) provided a good compromise. In early experiments with waveform icons, we also realized that, while it may be tempting to normalize the waveform locally to optimize the space allocated for the icon, global normalization was crucial to be able to understand the icons in the context of the plot. A formal user study would help confirm these findings.

8. Conclusions

Combining audio analysis and dimensionality reduction is a promising approach for designing novel music creation interfaces. For this purpose, we have introduced a general framework for modular combination of segmentation, analysis and projection to

two-dimensional spaces, including the representation of sound segments as icons. We have validated experimentally the potential of feature learning, the suitability of the UMAP algorithm, and the usefulness of the Hungarian algorithm to create grid layouts. Further evaluation is needed for several aspects, notably the use of icons and color.

We have implemented the framework in a SuperCollider library that can be used by researchers, musicians, and sound designers to experiment with these algorithms in music creation contexts. We hope that our work will enable further technical and artistic research on the use of unsupervised machine learning for music creation.

Author Contributions: Conceptualization, G.R., A.X., O.G. and P.A.T.; methodology, G.R.; software, G.R., A.X., O.G. and P.A.T.; data curation, G.R.; writing—original draft preparation, G.R.; writing—review and editing, G.R., A.X., O.G. and P.A.T.; visualization, G.R. and A.X.; project administration, P.A.T.; funding acquisition, P.A.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was part of the Fluid Corpus Manipulation project (FluCoMa), which has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 725899).

Data Availability Statement: The code for the Fluid Corpus Map library is available at https://github.com/flucoma/FluidCorpusMap2 (accessed on 1 November 2021). The code for the experiments is available at the companion website of this article: https://www.flucoma.org/APPLSCI/ (accessed on 1 November 2021). The data for the SOL dataset are available at https://zenodo.org/record/3740399 (accessed on 1 November 2021). The data for the Urban dataset are available from https://urbansounddataset.weebly.com/urbansound8k.html (accessed on 1 November 2021). The data for the Drums and Gaver datasets contain copyrighted materials and cannot be released. The code for the Fluid Corpus Manipulation Toolkit is publicly available at https://www.flucoma.org (accessed on 1 November 2021).

Acknowledgments: We would like to thank the reviewers for their time and thoughtful comments. Also, thanks to Claudine Levasseur for proofreading. Finally, we would like to thank the FluCoMa community for their engagement and feedback.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CCE	Creative Coding Environments
DAW	Digital Audio Workstations
FCMT	Fluid Corpus Manipulation Toolkit
FFT	Fast Fourier Transform
KNN	K-Nearest Neighbors
MDS	Multi-Dimensional Scaling
MFCC	Mel Frequency Cepstral Coefficients
MIR	Music Information Retrieval
MPEG-7	Moving Picture Experts Group standard 7
ODF	Onset Detection Functions
PC	Principal Components
PCA	Principal Components Analysis
SOM	Self-Organizing Maps
t-SNE	T-distributed Stochastic Network Embedding
UMAP	Uniform Manifold Approximation and Projection
VA	Variational Autoencoders

References

1. McCartney, J. Rethinking the computer music language: SuperCollider. *Comput. Music J.* **2002**, *26*, 61–68. [CrossRef]
2. Puckette, M. Max at seventeen. *Comput. Music J.* **2002**, *26*, 31–43. [CrossRef]
3. Lazzarini, V. The development of computer music programming systems. *J. New Music Res.* **2013**, *42*, 97–110. [CrossRef]

4. Dupont, S.; Ravet, T.; Picard-Limpens, C.; Frisson, C. Nonlinear Dimensionality Reduction Approaches Applied to Music and Textural Sounds. In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), San Jose, CA, USA, 15–19 July 2013; Zhang, J., Dan Schonfeld, D.D.F., Eds.; IEEE Computer Society: Los Alamitos, CA, USA, 2013; pp. 1–6.
5. Font, F.; Bandiera, G. Freesound Explorer: Make Music While Discovering Freesound. In Proceedings of the Web Audio Conference (WAC), London, UK, 21–23 August 2017; Thalmann, F., Ewert, S., Eds.; Queen Mary University of London: London, UK, 2017.
6. Leandro Garber, T.C.; Amusategui, J.C. Audiostellar, an Open Source Corpus-Based Musical Instrument for Latent Sound Structure Discovery and Sonic Experimentation. In Proceedings of the International Conference on Computer Music (ICMC), Santiago, Chile, 25–31 July 2021; Cádiz, R.F., Ed.; International Computer Music Association: San Francisco, CA, USA, 2021.
7. Grill, T.; Flexer, A. Visualization of Perceptual Qualities in Textural Sounds. In Proceedings of the International Conference on Computer Music (ICMC), Ljubljana, Slovenia, 9–14 September 2012; International Computer Music Association: San Francisco, CA, USA, 2012; pp. 589–596.
8. Roma, G.; Serra, X. Music Performance by Discovering Community Loops. In Proceedings of the Web Audio Conference (WAC), Paris, France, 26–28 January 2015; Goldszmidt, S., Schnell, N., Saiz, V., Matuszewski, B., Eds.; IRCAM: Paris, France, 2015.
9. Tremblay, P.A.; Roma, G.; Green, O. Digging It: Programmatic Data Mining as Musicking. In Proceedings of the International Conference on Computer Music (ICMC), Santiago, Chile, 25–31 July 2021; International Computer Music Association: San Francisco, CA, USA, 2021.
10. Coleman, G. Mused: Navigating the Personal Sample Library. In Proceedings of the International Conference on Computer Music (ICMC), Copenhagen, Denmark, 27–31 August 2007; Jensen, K.K., Serafin, S., Eds.; International Computer Music Association: San Francisco, CA, USA, 2007; pp. 324–327.
11. Schwarz, D.; Beller, G.; Verbrugghe, B.; Britton, S. Real-Time Corpus-Based Concatenative Synthesis With CataRT. In Proceedings of the International Conference on Digital Audio Effects (DAFx), Montreal, QC, Canada, 18–20 September 2006; Depalle, P., Verfaille, V., Eds.; McGill University: Montreal, QC, Canada, 2006; pp. 279–282.
12. McAdams, S. Perspectives on the contribution of timbre to musical structure. *Comput. Music J.* **1999**, *23*, 85–102. [CrossRef]
13. Esling, P.; Chemla-Romeu-Santos, A.; Bitton, A. Bridging Audio Analysis, Perception and Synthesis With Perceptually-Regularized Variational Timbre Spaces. In Proceedings of the International Society for Music Information Retrieval (ISMIR), Paris, France, 23–27 September 2018; pp. 175–181.
14. Pampalk, E.; Hlavac, P.; Herrera, P. Hierarchical Organization and Visualization of Drum Sample Libraries. In Proceedings of the International Conference on Digital Audio Effects (DAFx), Naples, Italy, 5–8 October 2004; pp. 378–383.
15. Heise, S.; Hlatky, M.; Loviscach, J. SoundTorch: Quick Browsing in Large Audio Collections. In Proceedings of the 125th Convention of the Audio Engineering Society, San Francisco, CA, USA, 2–5 October 2008.
16. Tenenbaum, J.B.; de Silva, V.; Langford, J.C. A global geometric framework for nonlinear dimensionality reduction. *Science* **2000**, *290*, 2319–2323. [CrossRef] [PubMed]
17. Fasciani, S.; Wyse, L.L. Adapting General Purpose Interfaces to Synthesis Engines Using Unsupervised Dimensionality Reduction Techniques and Inverse Mapping From Features to Parameters. In Proceedings of the International Conference on Computer Music (ICMC), Ljubljana, Slovenia, 9–14 September 2012; International Computer Music Association: San Francisco, CA, USA, 2012; pp. 467–472.
18. Fried, O.; DiVerdi, S.; Halber, M.; Sizikova, E.; Finkelstein, A. IsoMatch: Creating informative grid layouts. *Comput. Graph. Forum* **2015**, *34*, 155–166. [CrossRef]
19. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
20. McInnes, L.; Healy, J.; Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv* **2018**, arXiv:1802.03426.
21. Richan, E.; Rouat, J. A proposal and evaluation of new timbre visualization methods for audio sample browsers. *Pers. Ubiquit. Comput.* **2021**, *25*, 723–736. [CrossRef]
22. Tremblay, P.A.; Green, O.; Roma, G.; Harker, A. From Collections to Corpora: Exploring Sounds Through Fluid Decomposition. In Proceedings of the International Conference on Computer Music (ICMC), New York, NY, USA, 16–23 June 2019; International Computer Music Association: San Francisco, CA, USA, 2019.
23. Bello, J.P.; Daudet, L.; Abdallah, S.; Duxbury, C.; Davies, M.; Sandler, M.B. A tutorial on onset detection in music signals. *IEEE Trans. Audio Speech Lang. Process.* **2005**, *13*, 1035–1047. [CrossRef]
24. Foote, J. Automatic Audio Segmentation Using a Measure of Audio Novelty. In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), New York, NY, USA, 30 July–2 August 2000; IEEE Computer Society: Los Alamitos, CA, USA, 2000; pp. 452–455.
25. Coates, A.; Ng, A.Y. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*; Montavon, G., Orr, G.B., Müller, K.R., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 561–580.
26. Salamon, J.; Bello, J.P. Unsupervised Feature Learning for Urban Sound Classification. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, Australia, 19–24 April 2015; pp. 171–175.
27. Stowell, D.; Plumbley, M.D. Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning. *PeerJ* **2014**, *2*, e488. [CrossRef] [PubMed]

28. Ellis, D.P. Chroma Feature Analysis and Synthesis. 2007. Available online: https://www.ee.columbia.edu/~dpwe/resources/matlab/chroma-ansyn/ (accessed on 1 November 2021).
29. Kim, H.G.; Moreau, N.; Sikora, T. *MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval*; John Wiley & Sons: Hoboken, NJ, USA, 2006.
30. Xiong, Z.; Radhakrishnan, R.; Divakaran, A.; Huang, T.S. Comparing MFCC and MPEG-7 Audio Features For Feature Extraction, Maximum Likelihood HMM and Entropic Prior HMM for Sports Audio Classification. In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Hong Kong, China, 6–10 April 2003; Volume 5, p. V-628.
31. Brossier, P.M. Automatic Annotation of Musical Audio for Interactive Applications. Ph.D. Thesis, Queen Mary University of London, London, UK, 2006.
32. Grey, J.M.; Gordon, J.W. Perceptual effects of spectral modifications on musical timbres. *J. Acoust. Soc. Am.* **1978**, *63*, 1493–1500. [CrossRef]
33. Roma, G.; Green, O.; Tremblay, P.A. Adaptive Mapping of Sound Collections for Data-driven Musical Interfaces. In Proceedings of the 19th International Conference on New Interfaces for Musical Expression (NIME), Porto Alegre, Brazil, 3–6 June 2019; Queiroz, M., Xambó Sedó, A., Eds.; UFRGS: Porto Alegre, Brazil, 2019; pp. 313–318.
34. RasterFairy. 2015. Available online: https://github.com/Quasimondo/RasterFairy (accessed on 1 November 2021).
35. Jonker, R.; Volgenant, A. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* **1987**, *38*, 325–340. [CrossRef]
36. Bentley, J.L. Multidimensional binary search trees used for associative searching. *Commun. ACM* **1975**, *18*, 509–517. [CrossRef]
37. Schwarz, D.; Schnell, N.; Gulluni, S. Scalability in Content-Based Navigation of Sound Databases. In Proceedings of the International Conference on Computer Music (ICMC), Montreal, QC, Canada, 16–21 August 2009; International Computer Music Association: San Francisco, CA, USA, 2009; pp. 13–16.
38. Fried, O.; Jin, Z.; Oda, R.; Finkelstein, A. AudioQuilt: 2D Arrangements of Audio Samples using Metric Learning and Kernelized Sorting. In Proceedings of the International Conference on New Interfaces for Musical Expression, London, UK, 30 June–4 July 2014; Goldsmiths, University of London: London, UK, 2014; pp. 281–286.
39. Frisson, C.; Dupont, S.; Yvart, W.; Riche, N.; Siebert, X.; Dutoit, T. AudioMetro: Directing Search for Sound Designers through Content-Based Cues. In Proceedings of the 9th Audio Mostly: A Conference on Interaction With Sound, Aalborg, Denmark, 1–3 October 2014; Grimshaw, M., Walther-Hansen, M., Eds.; Association for Computing Machinery: New York, NY, USA, 2014.
40. Basalaj, W. *Proximity Visualisation of Abstract Data*; Technical Report UCAM-CL-TR-509; Computer Laboratory, University of Cambridge: Cambridge, UK, 2001.
41. Cella, C.E.; Ghisi, D.; Lostanlen, V.; Lévy, F.; Fineberg, J.; Maresz, Y. OrchideaSOL: A dataset of extended instrumental techniques for computer-aided orchestration. *arXiv* **2020**, arXiv:2007.00763.
42. Salamon, J.; Jacoby, C.; Bello, J.P. A Dataset and Taxonomy for Urban Sound Research. In Proceedings of the ACM International Conference on Multimedia (MM), Orlando, FL, USA, 3–7 November 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 1041–1044.
43. Roma, G.; Janer, J.; Kersten, S.; Schirosa, M.; Herrera, P.; Serra, X. Ecological acoustics perspective for content-based retrieval of environmental sounds. *EURASIP J. Audio Speech Music Proc.* **2010**, 960863. [CrossRef]
44. Gaver, W.W. What in the world do we hear? An ecological approach to auditory event perception. *Ecol. Psychol.* **1993**, *5*, 1–29. [CrossRef]
45. Van Der Maaten, L.; Postma, E.; Van den Herik, J. Dimensionality reduction: a comparative review. *J. Mach. Learn. Res.* **2009**, *10*, 13.
46. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
47. UMAP. 2018. Available online: https://github.com/lmcinnes/umap (accessed on 1 November 2021).
48. Performance Comparison of Dimension Reduction Implementations. 2018. Available online: https://umap-learn.readthedocs.io/en/latest/benchmarking.html (accessed on 1 November 2021).
49. Bostock, M.; Ogievetsky, V.; Heer, J. D^3 data-driven documents. *IEEE Trans. Vis. Comput. Graph.* **2011**, *17*, 2301–2309. [CrossRef] [PubMed]