Networked Music Performance in PatchXR and FluCoMa

Jonathan Bell

Networked Music Performance in PatchXR and FluCoMa

Jonathan Bell
PRISM-CNRS / XR2C2-CTEL
belljonathan50@gmail.com

ABSTRACT

The present study proposes to explore a sound corpus in VR, in which audio data is sliced and analysed in FluCoMa in order obtain relatively large collections of samples clustered by timbre similarity in a 3d room. The recent implementation of the multiplayer feature in PatchXR lets envisage wide variety of gesture-based control interfaces querying those corpora, and in which performers can interact remotely in order to simulate a chamber music situation.

1. INTRODUCTION

The recent emergence of multiplayer capabilities of a VR software such as patchXR [1] urges to find meaningful instrument design in order to remotely and collaboratively interact musically with a digital instrument, in multiplayer VR (or metaverse). In search of experiences that would relate to those found in traditional chamber music, the solution proposed here focuses on the exploration of a sound corpus projected on a 3d space, which the users can then navigate with his hand controllers (see Fig. 1): in an etude for piano and saxophone for instance [1] , one musician plays blue buttons (the saxophone samples), and the other the yellow buttons (the piano samples).

An important reason here for using VR to explore a 3D dataset is that it allows users to interact with the data in a more natural and immersive way compared to a 2d plane (Chapter 3 will show how the present study derives from the CataRT 2d interface project), using the experience both as a tool for performance as well as data visualisation and analysis. Users can move around and explore the data from different angles, which can help them to better understand the relationships between different data points and identify patterns, which becomes more evident as the number of points increases. The use of machine learning (dimensionality reduction in our case) renders a world in which the absolute coordinates of each point has no more link to the descriptor space (the high sounds cannot be mapped to the y axis for instance), but offers compelling results for clustering information relating to the different playing styles of the instrument that is being analysed: as an example, in this extract [2] based on flute sounds, the opening shows a clear opposition between two types of gestures: 1/ (0'00) staccato notes and 2/ (0'05) legato scale-like type of material. This contrast in timbral quality is made explicit by a movement of the avatar which jumps from a cluster of buttons to another.

[1] https://youtu.be/kIi7YdzP2Nw?t=89
[2] https://youtu.be/777fqIIJCY4

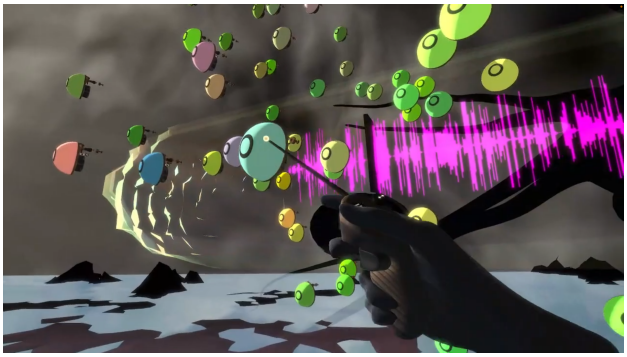

Figure 1. A VR interface in which each button in the world corresponds to a slice of the sound file. Machine learning helps bringing closer sounds that share common spectral characteristics.

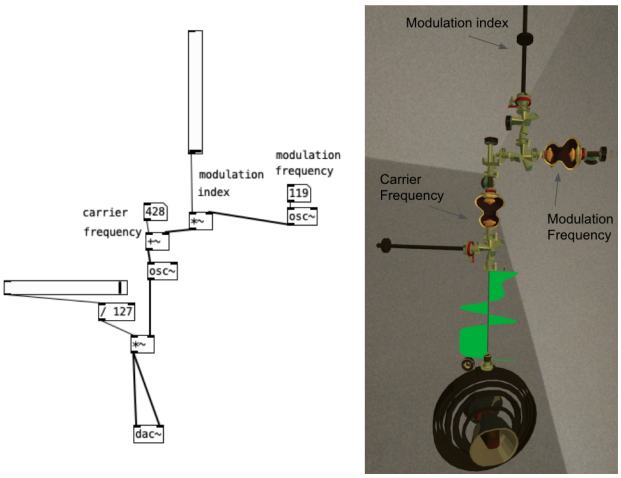

Figure 2. The implementation of the FM algorithm 1/ in Pure Data (left) 2/ in PatchXR (right).

2. VIRTUAL REALITY

2.1 Musical Metaverse

Turchet conducted a thorough study on what the musical metaverse means today [2], although the realm still is in its infancy. Berthaut [3] reviewed 3D interaction techniques and examined how they can be used for musical control. A survey of the emerging field of networked music performances in VR was offered by Loveridge [4]. Atherton and Wang [18] provided an overview of recent musical works

in VR, while Çamcı and Hamilton [5] identified research trends in the Musical XR field through a set of workshops focusing on Audio-first VR.

Amongst a plethora of tools available today, PatchXR has caught our attention primarily because its ressemblance to Max/Pure Data environments (see Fig. 2).

2.2 PatchXR

PatchXR [1] is a tool for creating immersive virtual reality experiences in which users can design and build interactive worlds, games, and music experiences. At its core, Patch is a modular synthesis platform that allows users to create and connect different building blocks, or "patches" together to create complex systems. These building blocks can include everything from 3D models and textures to physics simulations, lighting controls, and, most importantly, audio digital signal processing.

One of the key features of Patch is its ability to enable collaboration between users. Patches can be shared and remixed, allowing multiple users to work together on a single project or create something entirely new. In addition, Patch has a robust library of resources for users to draw from, including tutorials, documentation, and sample patches. The community around Patch is also very active, with regular competitions, events, and meet ups happening around the world.

One of the most exciting aspects of Patch is its potential for use in music performance and composition. The modular design of the platform allows users to create complex audio and visual environments that can be controlled in real-time, opening up new possibilities for live music and audiovisual performances. Patch has been used in a variety of contexts, from creating interactive installations and exhibits to developing VR training simulations and games. Its flexibility and modular design make it a powerful tool for anyone interested in exploring the creative possibilities of VR.

3. CORPUS-BASED CONCATENATIVE SOUND SYNTHESIS (CBCS)

Corpus-Based Concatenative Sound Synthesis (CBCS) is a technique used in computer music that involves constructing a sound or music piece by concatenating (joining together) smaller units of sound, such as phonemes in speech synthesis or musical phrases in music synthesis. It is used in our case to model an improvising instrumental musician by creating a database of recorded musical phrases or segments that can be combined and rearranged in real-time to create a musical performance that sounds like it is being improvised.

Today nearly 20 years old if one refers to the first CataRT publications [6], CBCS today enjoys an increasing popularity. Various apps today are based on similar principals (AudioStellar, Audioguide, LjudMAP or XO). The democratisation of audio analysis and machine learning tools such as the FluCoMa package (for Max, SuperCollider and Pure Data) encourages computer music practitioners to engage in this field at the crux between music creation and data science/machine learning.

3.1 Timbre Space

In spite of promising advances in the domain of deep learning applied to sound synthesis [7] [8], CBCS tools may earn their popularity from a metaphor which leads back to the early days of computer music: the notion of timbre space, developed by Wessel [9] and Grey [10], according to which the multi-dimentional qualities of *timbre* may be better understood using spatial metaphors (e.g. the timbre of the English horn being closer to basson than is it of trumpet).

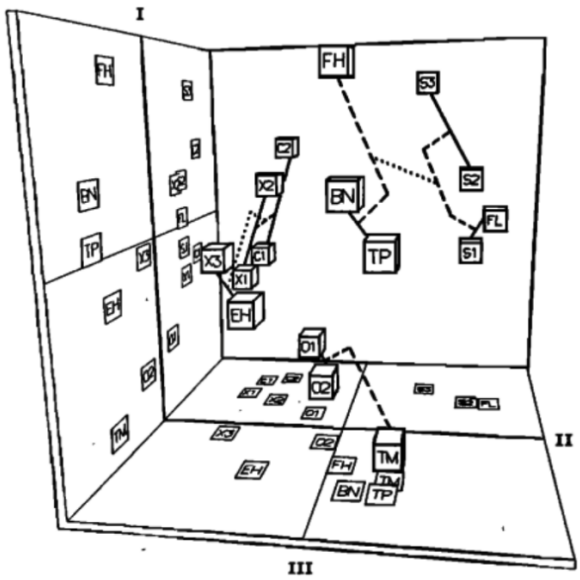

Figure 3. Multidimensional perceptual scaling of musical timbres (John M. Grey[10]). Sounds available at: https://muwiserver.synology.me/timbrespaces/grey.htm.

Pioneers in the perception of timbre studies such as Grey [10], J.C. Risset, D. Wessel, [11] or Stephen McAdams [12] [13] most often define timbre by underline what it is not. Risset and Wessel, for instance, define it as follow: *It is the perceptual attribute that enables us to distinguish among orchestral instruments that are playing the same pitch and are equally loud.* The co-variance such parameters (pitch, loudness and timbre), however, leads Schwarz to distinguish timbre space and CBCS notions: '*Note that this concept is similar but not equivalent to that of the timbre space put forward by Wessel and Grey [7, 24], since timbre is defined as those characteristics that serve to distinguish one sound from another, that remain after removing differences in loudness and pitch. Our sound space explicitly includes those differences that are very important to musical expression.*" [14]

The workflow described in Chapter 5 gave in practice strong evidence of inter-dependance between register, timbre and dynamics, particularly when the analysis run over a single instrument sound file (e.g. 30 minutes of solo flute), and chopped in short samples. The system will then precisely be able to find similarity between instrumental passages played in the same register, same dynamic, and same playing technique (e.g. a flute playing fast trills mezzo forte, in mid-low register, with air).

3.2 Corpus-Based Concatenative Synthesis - State of the art

A wide array of technologies today can be called corpus-based concatenative synthesis, in the sense that they allow, through segmentation and analysis, to explore large quantities of sound. Some of them are presented as "ready-made" solutions, such as the recent *Audiostellar* [15], or SCMIR [3] for SuperCollider. Hackbarth's AudioGuide [16] offers a slightly different focus because it uses the morphology/timeline of a soundfile to produce a concatenated output. Within the Max world finally, two environments appear as highly customizable: IRCAM's MuBu [17] and the more recent EU funded FluCoMa [18] project. CataRT is now fully integrated in MuBu, whose purpose encompasses multimodal audio analysis as well as machine for movement and gesture recognition [19]. This makes MuBu extremely general purpose, but also difficult to grasp. The data processing tools in MuBu are mostly exposed in the pipo plugin framework [20], which can compute for instance mfcc analysis on a given audio buffer [4] by embedding the pipo.mfcc plugin inside the mubu.process object.

FluComa also aims to be general purpose, but seems particularly suited to perform two popular specific tasks. With only limited knowledge of the framework nor of theory laying behind the algorithms it uses (such as those dimentionality reduction, mfcc analysis, or neural network training), the framework allows: 1/ to segment, analyse and represent/playback a sound corpus 2/ to train a neural network to control a synthesizer, in a manner reminiscent of Fiebrink's Wekinator [21].

Only the tools for segmentation, analysis, representation and playback (described in detail in Chapter 5) were used here, for they precisely fit the needs of corpus-based synthesis.

4. CONNECTING PATCHXR AND FLUCOMA

Porting analysis made in Max/FluCoMa to PatchXR consists in generating a world in which each sonic fragment's 3d position follows coordinates delivered by FluCoMa.

The structure of a .patch file (a patchXR world) follows the syntax of a .maxpat (for Max) or .pd file (for pure data) in the sense that it first declares the objects used, and then the connexions between them. This simple structure helped to generate a javascript routine generating a template world, taking as input 1/ dictionaries (json files) with each segment's 3d coordinates and 2/ each segment's temporal position in the sound file, and as output a new .patch file (a world accessible in VR, see general workflow on Fig. 4).

[3] A demo is available at: https://youtu.be/jxo4StjV0Cg

[4] MFCC stands for Mel-Frequency Cepstral Coefficients. It is a type of feature extraction method that is commonly used in speech and speaker recognition systems. MFCCs are used to represent the spectral characteristics of a sound in a compact form that is easier to analyze and process than the raw waveform. They are calculated by applying a series of transformations to the power spectrum of a sound signal, including a Mel-scale warping of the frequency axis, taking the logarithm of the power spectrum, and applying a discrete cosine transform (DCT) to the resulting coefficients. The resulting coefficients, which are called MFCCs, capture the spectral characteristics of the sound and are commonly used as features for training machine learning models for tasks such as speech recognition and speaker identification.

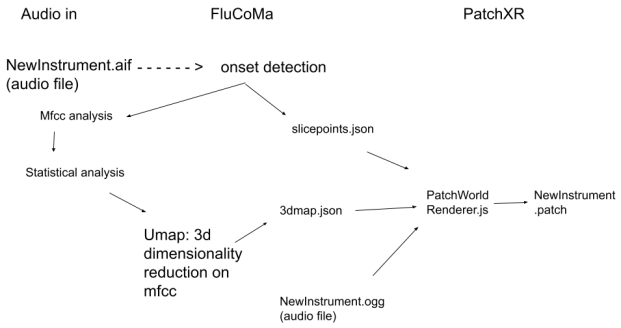

Figure 4. General workflow: from an input audio file to its .patch 3d representation in PatchXR.

After a few trials in which the x y z coordinates of a world directly represented audio descriptors such as loudness, pitch and centroid [5] , I more systematically used mfcc analysis and dimensionality reduction, as will be shown in section 5.

Section 6 will present different javascript programs and Max patches that were developed in order to diversify the ways in which FluCoMa analysis is represented in PatchXR, and how the user can interact with it.

5. WORKFLOW - ANALYSIS IN FLUCOMA

My experiments have focussed on musical instrument corpora almost exclusively [6] . The tools presented here can efficiently generate plausible virtuosic instrumental music but recent uses found more satisfying results in slower, quieter, "Feldman-like" types of textures. Various limitations on the playback side (either in standalone VR, or on a Pure Data sampler for RaspberryPi described in Section 6.2) have imposed restrictions in the first stages on the amount of data it could handle (less than 5 minutes in AIFF in PatchXR) or the number of slice the sample could be chunked into (256 because of limitation of lists in Max, a limitation that has also been surpassed since). Both limitations were later overcome (use of the compressed ogg format in PatchXR, and taking advantage of longer sound files since version 672, increase of internal buffer size in fluid.buf2list in FluCoMa), thus allowing for far more convincing models.

Using concatenative synthesis to model an improvising instrumental musician typically involves several steps:

1. Segmentation of a large soundfile: This involves dividing a large audio recording of the musician's performance into smaller units or segments.
2. Analysis: These segments are then organised in a database according to various descriptor data (mfcc in our case).
3. Scaling/pre-processing: scaling is applied for better visualisation.
4. Dimension reduction: Based on mfcc descriptors, the dimensionality of the data is reduced in order to

[5] https://youtu.be/1LHcbYh2KCI?t=19

[6] For cello: https://youtu.be/L-MiKmsIzjM For various instruments: https://www.youtube.com/playlist?list=PLc_WX6wY4JtnNqu4Lwe2YzEUq9S1IMvUk For flute: https://www.youtube.com/playlist?list=PLc_WX6wY4JtlbjLuLHDZhlx78sTDn

make it more manageable and easier to work with. This can be done using techniques such as principal component analysis (PCA) singular value decomposition (SVD), or Uniform Manifold Approximation and Projection (UMAP, preferred in our case).

5. Near neighbours sequencing: Once the segments have been organised and analysed, the algorythm selects and combines them in real-time based on certain input parameters or rules to create a simulated musical performance that sounds like it is being improvised by the musician. We use here a near neighbours algorithm, which selects segments that are similar in some way (e.g., in terms of pitch, loudness, or timbre - thanks to similarities revealed by umap on mfccs in our case) to the current segment being played.

We will now describe these steps in further detail:

5.1 Slicing

Slicing a sound file musically allow various possible exploitations in the realm of CBCS. In MuBu onset detection is done with pipo.onseg or pipo.gate. FluCoMa expose five different onset detection algorithms:

1. fluid.ampslice: Amplitude-based detrending slicer
2. fluid.ampgate: Gate detection on a signal
3. fluid.onsetslice: Spectral difference-based audio buffer slicer
4. fluid.noveltyslice: Based on self-similarity matrix (SSM)
5. fluid.transcientslice: Implements a de-clicking algorithm

Onsetslice only was extensively tested. The only tweaked parameters were a straight-forward "threshold" as well as a "minslicelength" argument, determining the shortest slice allowed (or minimum duration of a slice) in hopSize. This introduce a common limitation in CBCS: the system strongly biases the user to choose short samples for better analysis results, and more interactivity, when controlling the database with a gesture follower. Aaron Einbond remarks in the use of CataRT how short samples most suited his intention: "*Short samples containing rapid, dry attacks, such as close-miked key-clicks, were especially suitable for a convincing impression of motion of the single WFS source. The effect is that of a virtual instrument moving through the concert hall in tandem with changes in its timbral content, realizing Wessel's initial proposal.*"[22]

A related limitation of concatenative synthesis lies in the fact that short samples will demonstrate the efficiency of the algorithm [7], but at the same time moves away from the "plausible simulation" sought in the present study. A balance therefore must be found between the freedom imposed by large samples, and the refined control one can obtain with short samples.

A direct concatenation of slices clicks in most cases on the edit point, which can be avoided through the use of ramps. The second most noticeable glitch on concatenation concerns the interruption of low register resonances, which even a large reverb fails making sound plausible. Having a low threshold and large "minslicelenght" results in equidistant slices, all of identical durations, as would do the pipo.onseg object in MuBu.

Because we listen to sound in time, this parameter responsible for the *duration of samples* is of prior importance.

5.2 MFCC on each slice - across one whole slice/segment

Multidimensional MFCC analysis: MFCC (Mel-Frequency Cepstral Coefficient) analysis is a technique used to extract features from audio signals that are relevant for speech and music recognition. It involves calculating a set of coefficients that represent the spectral envelope of the audio signal, or decomposing a sound signal into a set of frequency bands and representing the power spectrum of each band with a set of coefficients. The resulting MFCC coefficients capture important spectral characteristics of the sound signal (albeit hardly interpretable by the novice user), such as the frequency and magnitude of the spectral peaks. We will see that combines with *umap*, it is able to capture the spectral characteristics of the musician's playing style.

5.3 Statistical Analysis Over Each Slice

BufStats is used to calculate statistical measures on data stored in a buffer channel. BufStats calculates seven statistics on the data in the buffer channel: mean, standard deviation, skewness, kurtosis, low, middle, and high values. These statistics provide information about the central tendency of the data and how it is distributed around that tendency. In addition to calculating statistics on the original buffer channel, BufStats can also calculate statistics on up to two derivatives of the original data, apply weights to the data using a weights buffer, and identify and remove outlier frames. These statistical measures can be useful for comparing different time-series data, even if they have different length, and may provide better distinction between data points when used in training or analysis. The output of BufStats is a buffer with the same number of channels as the original data, with each channel containing the statistics for its corresponding data in the original buffer.

5.4 Normalization

The FluCoMa package proposes several scaling or preprocessing tools, amongst which normalization and standardization were used. Standardization and normalization are techniques used to transform variables so that they can be compared or combined in statistical analyses. Both techniques are used to make data more comparable, but they work in slightly different ways.

Standardization scales a variable to have a mean of 0 and a standard deviation of 1, while normalization scales a variable to have a minimum value of 0 and a maximum value of 1. Normalization ccaling was found easier to use both in 2-D (in FluCoMa, the fluid.plotter object), as well as in the VR 3D world in which the origin corresponds to a corner of the world. The fluid.nomalize object features an "@max" attribute (1 by default), which then maps directly to the dimensions of the VR world.

[7] e.g. https://youtu.be/LD0ivjyuqMA?t=3032

5.5 Dimensionality Reduction - UMAP

Dimensionality reduction is a technique used in machine learning to reduce the number of features (dimensions) in a dataset. The goal of dimensionality reduction is to simplify the data without losing too much information. Various dimensionality reduction algorithms are presented in an early FluCoMa study [23], with interestingly no mention of UMAP, later favoured.

UMAP (Uniform Manifold Approximation and Projection) is a non-linear dimensionality reduction technique that is based on the principles of topological data analysis. It can be used to visualize high-dimensional data in a lower-dimensional space. When applied to sound data analysed with MFCC (Mel-Frequency Cepstral Coefficients), UMAP reduces the dimensionality of the data and creates a visual representation of the sound in a 2- or 3-dimensional space.

By applying UMAP to the MFCC coefficients of a sound signal, it is possible to create a visual representation of the sound that preserves the relationships between the different MFCC coefficients (see Fig. 5).

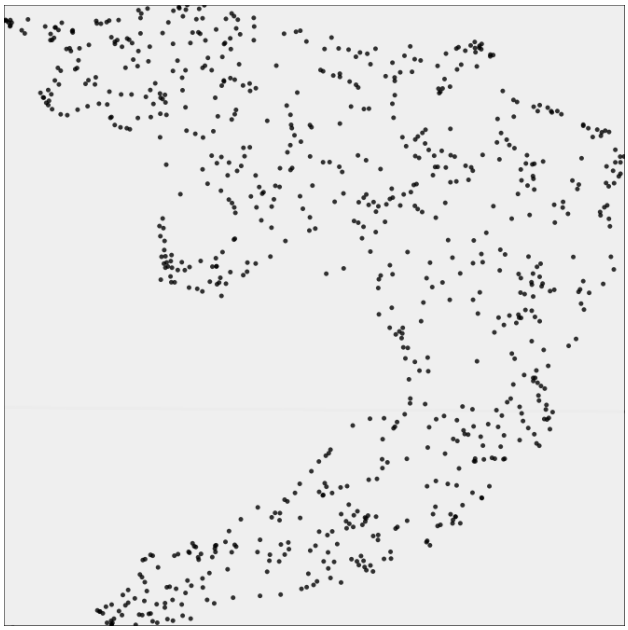

Figure 5. Dimensionality reduction of MFCCs help revealing spectral similarities. UMAP outputs coordinates in 2d or 3d.

UMAP is therefore used for its clustering abilities in the first place, helping for classification purposes. It helps identifying patterns or trends that may not be evident from the raw data. This can be useful for tasks such as exploring the structure of a sound dataset, identifying patterns or trends in the data, and comparing different sounds.

Most importantly, the non-linear dimensions proposed by UMAP (whether in 2d in Max or in 3 dimensions in PatchXR, and when compared to linear analyses in which, for instance, x, y and z correspond to pitch, loudness and centroid) gave far more “intelligent” clustering than more conventional parameter-consistent types of representations.

5.6 Neighbourhood queries

The neighbourhood retrieval function is based in FluCoMa on K-d trees and the knn algorythm. In MuBu, the mubu.knn object serves similar tasks. The ml.kdtree object in the ml.star library [24] gives comparable results.

K-d trees (short for ”k-dimensional trees”) and k-nearest neighbours (k-NN) are two algorithms that are related to each other, but serve different purposes, a k-d tree is a data structure that is used to store and efficiently query a set of points in a k-dimensional space, while the k-NN algorithm is a machine learning algorithm that is used for classification or regression. Both algorithms are often used in applications such as pattern recognition, image classification, and data mining.

6. WORKFLOW IN PATCHXR

I have most often used FluCoMa and PatchXR to generate monophonic instruments (one performer plays one instrument at a time), most typically in experiences where players the players face one another [8] . In the case of ”button worlds” such as this one or those described in section 6.1, there is no need of nearest neighbour retrieval since the performer clicks exactly on the data point, and he (mediated by his avatar) reproduces what knn would do with an automated instrument: he will privilege in his choice the samples he can reach at hand, rather than constantly jump large distances between items (see Fig. 1).

In the worlds developed in Sections 6.2 and 6.3 on the other hand, data points are not explicitely represented and some near neighbour strategies need to be implemented. PatchXR exposes a wide range of blocks (a block corresponds to an object in Max or Pure Data) making it simple to access gesture data such as:

- The position/distance between hands/controllers and a reference.
- The rotation angles (x y z) of both hands’ controllers
- 2-d touchscreen-like controllers, where the user moves the xy position of a selector across a plane by manually grabbing it.
- 2-d lazer-like controllers, where the user moves the xy position of a selector remotely, as if using a lazer pointer towards a remote screen or board.
- 2-d pads, which allow to access the velocity at which the pad is hit
- 3-d slider or theremine like controllers, where the user moves the xyz position of a selector across a plane by manually grabbing it.
- A block called “interaction box” similar to 3-d slider, with the main difference that the user does not grab the selector, but instead comes in and out of the interactive zone
- 1-d sliders, knobs, buttons...

One of the current challenges consists in diversifying the ways in which the corpus is queried.

[8] https://youtu.be/WhuqOOuzzBw

One to one mapping of UMAP results such as those of section 6.1 use buttons facing each other, in order to prompt the players to face each other [9] .

When playing alone and controlling at the same time many instruments (the one-man-orchestra), encourages to use higher level type of control over automata, i.e. to implement the simple ability to concatenate automatically: play the next sample as soon as the previous one has stopped (see section 6.2).

6.1 One to one mapping between data points and buttons

The first javascript routine developed was designed to simply map a data point in the sonic space to a button in the virtual 3d space. By iterating over an array, the routine generates a world in which each button's coordinates are dictated by the FluCoMa umap analysis descripbed in section 5.5. Albeit simpler than the method that will be exposed later, this simple one to one to one mapping has several advantages, most importantly the haptic and visual feedback the performer gets when hitting each button.

6.2 Max/pd dependence

The second stage investigated the possibility to render the sounds on an array of raspberry pi computers [25]. While this method shows advantages in terms of patching (because of the convenience of using max and pure data), the main drawback is that patches designed in this way cannot be accessed by the PatchXR community. Documentation, similarly is harder to record since the sound is produced outside of PatchXR. The haptic and visual feedback is very different here in the sense that the user controls primarily the region of the space to be played, and when to start and stop playing (when his hand touches the interface or not). The way he plays is less rhythmical than in the button interface where the player "hits" [10] each sample (section6.1). Here on the contrary, the automat simply keeps playing as long as he touches the interface [11]

Flucoma exposes the fluid.kdtree that is able to find the k nearest number once it is given as input the coordinates of each data point (see Fig. 6). This method proved more suitable to control automata in which the player selects a region of a 2d plane together with the number of neighbours he wants the automate to improvise with.

Most satisfying results were achieved by sending messages to each RaspberriPi independently, according to its specific (static) IP address, with a simple syntax of a 2-integer list corresponding to: 1/which buffer to lookup 2/ which slice in this buffer to play, each Pi/speaker thus being able to play each sound in a pure data patch, in which each slice looks up an array with the corresponding slice points.

6.3 Nearest neighbour in PatchXR

The currently most frequently used program is a javascript routine that generates a world (a .patch file) in which the x y z coordinates of each data point are stored in a "knob board" block (the long rectangle in Fig.7). To measure the distance in 3D using Thales' theorem (or so called distance formula), we need to find the distance between two points in three-dimensional space, i.e find the diameter of the sphere that passes through both points.

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Fig. 8 shows the corresponding implementation in patchXR.

This fragment of visual program (called "abstraction" or "subpatch" in Max, and called "group" in PatchXR) is then used in a more complex patch which iterates (100 times per second, for instance) over an array (the rectangular "knob-board" object in Fig.7, so as to output the index (the value 234 in the figure) of the point situated the closest to the controller.

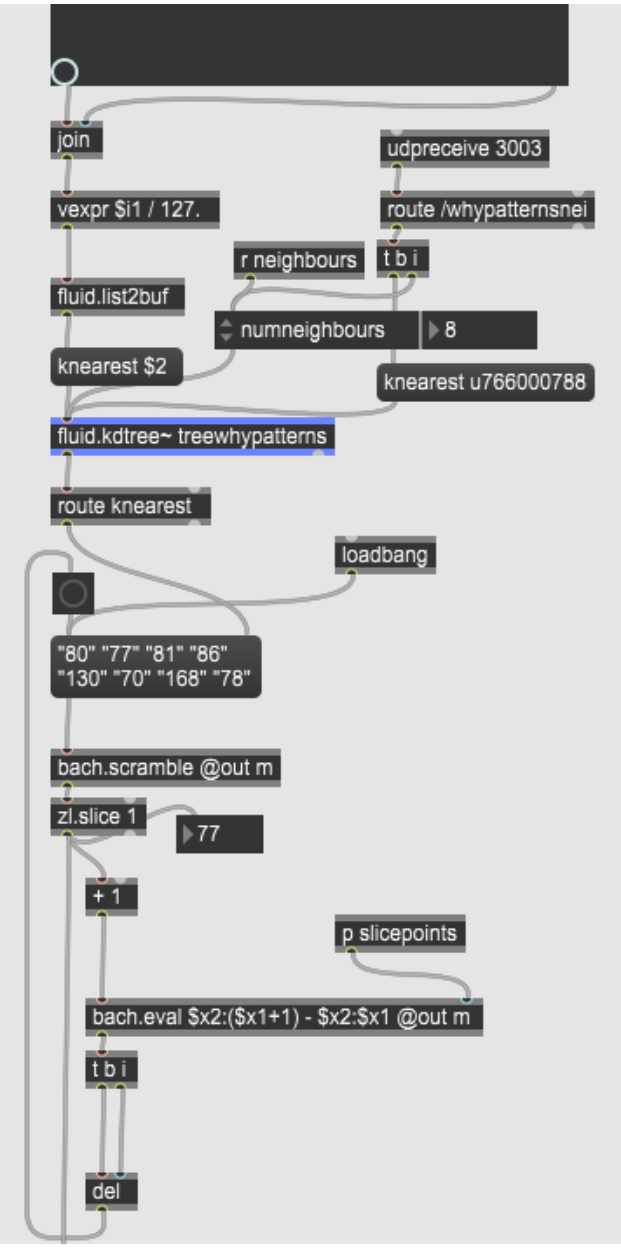

Figure 6. The *fluid.kdtree* object is used here to retrieve the 8 nearest neighbours of a point in a 2-d space.

[9] https://youtu.be/

[10] the opening of this video aptly conveys how the energy transfers form the player's gesture https://www.youtube.com/watch?v=glFdzbAJVRU

[11] https://youtu.be/vtob96F9cQw

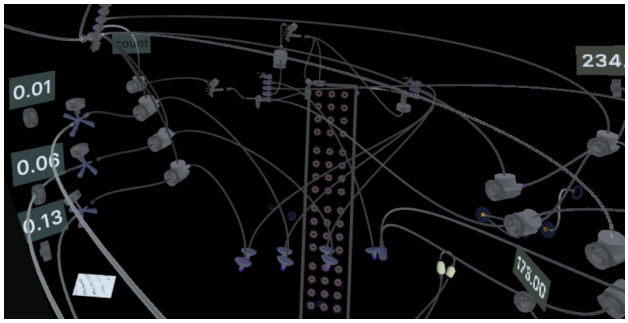

Figure 7. The coordinates of the player's controller (here 0.01, 0.06, 0.13) yield as a result index 234 as nearest neighbour (read from left to right).

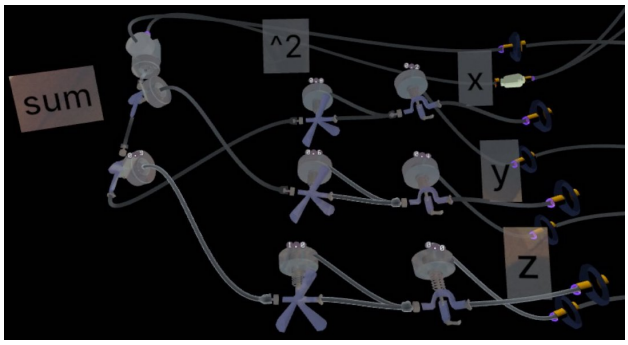

Figure 8. The implementation of Thales' theorem in patchXR (read from right to left).

7. NETWORKED MUSIC PERFORMANCE

The concept of online virtual collaboration has gained significant attention, notably through Meta's promotion in 2021. The application of this concept within the realm of "tele-improvisations" [26], most commonly referred to as Networked Music Performance (NMP) or holds the potential to overcome what was hitherto viewed as intrinsic limitation of the field, both from practitioners and from an audience point of view.

NMPs have indeed stimulated considerable research and experimentation whilst facing resistance at the same time. In his article "Not Being There", Miller Puckette argues that he finds "Having seen a lot of networked performances of one sort or another, I find myself most excited by the potential of networked "telepresence" as an aid to rehearsal, not performance." [27]. In *Embodiment and Disembodiment in NMP* [28], Georg Hajdu identifies an issue with a lack of readability from an audience perspective, who cannot perceive the gesture to sound relationship as would be the case in a normal concert situation: "These performances take machine–performer–spectator interactions into consideration, which, to a great deal, rely on embodied cognition and the sense of causality[...]. Classical cause-and-effect relationships (which also permeate the 'genuine' musical sign of the index) are replaced by plausibility, that is the amount to which performers and spectators are capable of 'buying' the outcome of a performance by building mental maps of the interaction." Performances of laptop orchestras, along with various other experiments using technology collaboratively, wether in local or distributed settings, have reported similar concerns, most commonly expressing a lack of embodiment in the performance. Although in its infancy, a first live performance staged/composed by the author in Paris, with participants distributed across Europe, showed a promising potential for tackling these issues, most importantly through its attempt at dramatising the use of avatars.

- JIM 23: https://youtu.be/npyfwqN02qE

A lot remains to be improved in order to give the audience an experience aesthetically comparable to that of the concert hall. Carefully orchestrated movements of cameras around the avatars, and faithful translation the headset experience of spatial audio with immersive visuals need further exploration, but would go beyond the scope of the present article.

8. CONCLUSIONS

We've proposed a workflow for corpus-based concatenative synthesis CBCS in multiplayer VR (or metaverse), arguing that machine learning tools for data visualisation offer revealing and exploitable information about the timbral quality of the material that is being analysed. In a wider sense, the present approach can be understood as reflexive practice on new media, according to which the notion of data-base may be considered an art form [29]).

The discussed tools for "machine listening" (FluCoMa, MuBu) help building intelligent instruments with relatively small amounts of data, the duration of samples appear crucial in CBCS. A balance must be found between 1/ short duration sample analysis which are easier to process and categorise and 2/ long samples which sound more natural in the context instrument-based simulations.

Acknowledgments

I am grateful for the support of UCA/CTEL, whose artist residency research program has allowed to hold these experiments, and for the support of PRISM-CNRS.

9. REFERENCES

[1] Andersson, "immersive audio programming in a virtual reality sandbox," *journal of the audio engineering society*, march 2019.

[2] L. Turchet, "Musical Metaverse: vision, opportunities, and challenges," *Personal and Ubiquitous Computing*, 01 2023.

[3] F. Berthaut, "3D interaction techniques for musical expression," *Journal of New Music Research*, vol. 49, no. 1, pp. 60–72, 2020.

[4] B. Loveridge, "Networked music performance in virtual reality: current perspectives," *Journal of Network Music and Arts*, vol. 2, no. 1, p. 2, 2020.

[5] A. Çamcı and R. Hamilton, "Audio-first VR: new perspectives on musical experiences in virtual environments," *Journal of New Music Research*, vol. 49, no. 1, pp. 1–7, 2020.

[6] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, "Real-Time Corpus-Based Concatenative Synthesis with CataRT," in *9th International Conference on Digital Audio Effects (DAFx)*, Montreal, Canada, Sep. 2006, pp. 279–282, cote interne IRCAM: Schwarz06c. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01161358

[7] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, *Deep Learning Techniques for Music Generation – A Survey*, Aug. 2019. [Online]. Available: https://hal.sorbonne-universite.fr/hal-01660772

[8] P. Esling, A. Chemla-Romeu-Santos, and A. Bitton, "Generative timbre spaces with variational audio synthesis," *CoRR*, vol. abs/1805.08501, 2018. [Online]. Available: http://arxiv.org/abs/1805.08501

[9] D. L. Wessel, "Timbre Space as a Musical Control Structure," *Computer Music Journal*, vol. 3, no. 2, pp. 45–52, 1979. [Online]. Available: http://www.jstor.org/stable/3680283

[10] K. Fitz, M. Burk, and M. McKinney, "Multidimensional perceptual scaling of musical timbre by hearing-impaired listeners," *The Journal of the Acoustical Society of America*, vol. 125, p. 2633, 05 2009.

[11] J.-C. Risset and D. Wessel, "Exploration of timbre by analysis and synthesis," *Psychology of Music*, pp. 113–169, 1999.

[12] S. Mcadams, S. Winsberg, S. Donnadieu, G. De Soete, and J. Krimphoff, "Perceptual scaling of synthesized musical timbres: Common dimensions, specificities, and latent subject classes," *Psychological research*, vol. 58, pp. 177–92, 02 1995.

[13] A. Caclin, S. Mcadams, B. Smith, and S. Winsberg, "Acoustic correlates of timbre space dimensions: A confirmatory study using synthetic tones," *The Journal of the Acoustical Society of America*, vol. 118, pp. 471–82, 08 2005.

[14] D. Schwarz, "The Sound Space as Musical Instrument: Playing Corpus-Based Concatenative Synthesis," in *New Interfaces for Musical Expression (NIME)*, Ann Arbour, United States, May 2012, pp. 250–253, cote interne IRCAM: Schwarz12a. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01161442

[15] L. Garber, T. Ciccola, and J. C. Amusategui, "AudioStellar, an open source corpus-based musical instrument for latent sound structure discovery and sonic experimentation," 12 2020.

[16] B. Hackbarth, N. Schnell, P. Esling, and D. Schwarz, "Composing Morphology: Concatenative Synthesis as an Intuitive Medium for Prescribing Sound in Time," *Contemporary Music Review*, vol. 32, no. 1, pp. 49–59, 2013. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01577895

[17] N. Schnell, A. Roebel, D. Schwarz, G. Peeters, and R. Borghesi, "MUBU and FRIENDS -ASSEMBLING TOOLS FOR CONTENT BASED REAL-TIME INTERACTIVE AUDIO PROCESSING IN MAX/MSP," *Proceedings of the International Computer Music Conference (ICMC 2009)*, 01 2009.

[18] P. A. Tremblay, G. Roma, and O. Green, "Enabling Programmatic Data Mining as Musicking: The Fluid Corpus Manipulation Toolkit," *Computer Music Journal*, vol. 45, no. 2, pp. 9–23, 06 2021. [Online]. Available: https://doi.org/10.1162/comj_a_00600

[19] F. Bevilacqua and R. Müller, "A Gesture follower for performing arts," 05 2005.

[20] N. Schnell, D. Schwarz, J. Larralde, and R. Borghesi, "PiPo, a Plugin Interface for Afferent Data Stream Processing Operators," in *International Society for Music Information Retrieval Conference*, 2017.

[21] R. Fiebrink and P. Cook, "The Wekinator: A System for Real-time, Interactive Machine Learning in Music," *Proceedings of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)*, 01 2010.

[22] A. Einbond and D. Schwarz, "Spatializing Timbre With Corpus-Based Concatenative Synthesis," 06 2010.

[23] G. Roma, O. Green, and P. A. Tremblay, "Adaptive Mapping of Sound Collections for Data-driven Musical Interfaces," in *New Interfaces for Musical Expression*, 2019.

[24] B. D. Smith and G. E. Garnett, "Unsupervised Play: Machine Learning Toolkit for Max," in *New Interfaces for Musical Expression*, 2012.

[25] *≪ PrÉ ≫ : connected polyphonic immersion*. Zenodo, Jul. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6806324

[26] R. Mills, "Tele-Improvisation: Intercultural Interaction in the Online Global Music Jam Session," in *Springer Series on Cultural Computing*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:57428481

[27] M. Puckette, "Not Being There," *Contemporary Music Review*, vol. 28, no. 4-5, pp. 409–412, 2009. [Online]. Available: https://doi.org/10.1080/07494460903422354

[28] G. Hajdu, "Embodiment and disembodiment in networked music performance," 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:149523160

[29] L. Manovich, "Database as Symbolic Form," *Convergence: The International Journal of Research into New Media Technologies*, vol. 5, pp. 80 – 99, 1999.