Human and Artificial Intelligence Alignment: AI as Musical Assistant and Collaborator

Ted Moore

May 26, 2021

Contents

List of Figures

1 Introduction

Electronic music composers have witnessed several critical technological revolutions in the field's short history. The affordability of analog tape allowed creators to manipulate recordings in new ways in the 1950s, launching the genre *musique concrète*. Extensive research into synthesizers in the 1960s offered new means of creating and sculpting sound, which directed the ethos of pop music in the 1970s and '80s. In the 1990s, personal computers became fast enough to process digital audio in real-time, leading to a wave of new works using live manipulation of sound from performing instrumentalists. Each revolution was launched by advancements in technology that empowered large numbers of composers to easily incorporate these tools into their creative processes. The technological revolution that electronic music is currently experiencing is the use of data science and machine learning.

This research outlines my thinking behind and implementations of artificial intelligence, machine learning,[1] and data science tools in my compositional and improvisational practice. One reason I am interested in using these technologies is the same reason they are finding uses across many disciplines and industries: automation. Ge Wang, in his article *Humans in the Loop: The Design of Interactive AI Systems*, reflects on what it means to consider the role of automation in the context of art, saying, "when we imagine 'automating' a pursuit like music making, we're forced to balance the *product* of work with something deeper — the meaning we derive from the *process* of doing it." (Wang 2019) As he suggests, I agree that the goal is not to remove myself from the process of art making or to fully "automate" the compositional or creative process. However, in my experience, the use of AI has not forced a "balancing" between product and process. Using AI has not minimized the role or meaning of process (or its duration–quite the opposite I believe), but has changed what my process is (a sentiment echoed by AI and creativity researcher Rebecca Fiebrink (CeReNeM 2019)). While using AI in my creative work, my process is comprised of activities that stimulate my creative thinking and feed my energy in ways that other strategies do not. In addition to pursuing this excitement, another goal of developing new and idiosyncratic processes with AI is the belief they will lead to new sounds, forms, compositional conceits–or more generally, new music–thereby expanding, advancing, and individuating my artistic voice.

Even with artificial intelligence at the center of these goals, it is not true that AI is at the center of all my work; most creative decisions and tasks are still carried out manually and many are in collaboration with an AI system. Wang explains the value in this approach, and clarifies his title: *Humans*

[1] Throughout this paper, as will become more clear, machine learning is considered a subcategory of artificial intelligence generally containing algorithms that improve their predictive accuracy through iterative training or trial and error.

in the Loop, saying, “It’s clear there is something worth preserving in many of the things we do in life, which is why automation can’t be reduced to a simple binary between ‘manual’ and ‘automatic.’ Instead, it’s about searching for the right balance between aspects that we would find useful to automate, versus tasks in which it might remain meaningful for us to participate.” (Wang 2019) Finding this balance has been the process of much of the research I present in Section 3, in particular the avoidance of a common AI pitfall: throwing data at a problem or idea and hoping AI will “figure it out”. Instead of dreaming for this “total automation”, I have learned to approach questions and challenges by starting from my artistic goals and identifying what AI can do (i.e., what can or should be automated) to help achieve them more quickly, successfully, interestingly, enjoyably, etc.

Pursuing these meta-goals has clarified two broad frameworks for how I think about implementing artificial intelligence in my work: (1) AI as assistant and (2) AI as collaborator. Assistants carry out tasks in prescribed, predictable ways, enabling humans in the performance or process (i.e., “*Humans in the Loop*”) to focus on other creative parameters. This framework most closely resembles AI automation as it is used in other industries or disciplines. To use Wang’s words, creating an AI *assistant* consists of identifying the “aspects that [one finds] useful to automate” and separating those from “tasks in which it might remain meaningful...to participate.” (Wang 2019) While technology (and the automation it provides) has generally been seen in an “assistant” capacity (or even labeled, such as the “Personal Digital Assistant”), my use of “assistant” represents something more specific: a more intelligent agent performing a more complex task usually involving data processing. Because of the complexity of the task, the AI must be *trained* (machine learning terminology used very intentionally here), similar to how a human supervisor might train a human *assistant*.

AI *collaborators*, which are given more attention in this paper, are more complex systems, resisting predictability, and are capable of “surprising” human users during the creative process with sounds, forms, gestures, connections, or other creative ideas not readily apparent to the human. These surprises or “creative suggestions” offered by artificially intelligent collaborators can then be responded to, curated, further explored, or denied, depending on the artistic moment or goal. Visual artist Mario Klingemann describes this process as he sorts through thousands of pictures created from Generative Adversarial Neural Networks to select the ones determined aesthetically pleasing (to him). (Simonite 2017) The AI collaborator, trained on thousands of images chosen by Klingemann, offers creative suggestions from within the learned parameter space to choose from. The “collaborator” framework less resembles “automation” as it might be viewed in other disciplines, but instead resembles the related, but more

Assistant	Collaborator
perceived as a "solo" performance	perceived as a "duet" performance
less time delay and/or distortion of human inputs	sufficiently distinct from human inputs
extends or elaborates human outputs	outputs separate from human actions
repeatable results	surprising results
elicits third person descriptions	elicits first person descriptions
control	contingency

Figure 1: A list of descriptors characterizing assistants and collaborators

agential, "automaton". As will be seen in Sections 1.1 and 2.2.1, the emergence of agency in these AI collaborators does not require machine learning algorithms; the interest in agential technologies has been strong in electronic music since its conception and continues to be a prevalent tool used by electronic musicians. A chart of some descriptions of assistants and collaborators can be seen in Figure 1.

This paper first outlines the differences in how assistants and collaborators are perceived, identified, and experienced. These comparisons are not intended to draw clear categorizations or even opposing ends of a continuum, but rather focus on how these frameworks resonate with the experience of using music technologies. Section 2 more closely analyzes how perceptions of agency and agents emerge from the use of these technologies, pointing towards a phenomenological definition of artificial intelligence in this context. Section 3 details four recent projects where I employ music technologies as assistants and collaborators. Finally, Section 4 concludes by considering how music technology tools can align with composers' values in different ways for different purposes.

1.1 The many faces of Assistants and Collaborators

The concepts of assistant and collaborator appear with different names in many writers' thinking about the role of technology in music and creativity. These labels are not intended to be categorical or even on opposing ends of one continuum. They overlap and engage with other concepts in ways that blur, but also deepen, their definitions. A few examples are interpreted here to elaborate my use of these terms and their relation to other research.

Solo vs. Duet In 1991, Robert Rowe offered a few classification systems for interactive music technologies, the most relevant of which are the "instrument" and "player" paradigms. (Rowe 1993) He describes "instruments" as being "concerned with constructing an extended musical instrument: performance gestures from a human player are analyzed by the computer and guide an elaborated output exceeding normal instrument response. Imagining such a system being played by a single performer, the musical result would be thought of as a solo," which is contrasted with, "Systems following a player paradigm try to construct an artificial player, a musical presence with a personality and behavior of its own, though it may vary in the degree to which it follows the lead of a human partner. A player paradigm system played by a single human would produce an output more like a duet." (Rowe 1993) The most interesting distinction he makes between "instrument" and "player" is the perception of a performance as being a solo or duet, which can be a useful way of identifying assistants and collaborators. During a "solo", the technology used is only *assisting* the performer; clearly it is doing some work in producing the music, but is not perceived as separate or agential. During a "duet", the work being done by the technology is separate from the human, therefore perceived as a collaborator. Rowe implies that the operative perceiver in making this distinction is the *audience*, however in my analyses (and most other descriptions) it is the perception of agency by the *composer/designer/performer* that identifies technology as a "collaborator". The process of perceiving agency is explored in Section 2.2.1.

Perception of Musical Intentionality In order to identify whether someone is perceiving a technology as an assistant or collaborator, one can pay attention to the way the technology is described. Marc Leman, offers a useful distinction in his chapter on musical intentionality, saying, "third-person descriptions are about repeatable measurements of phenomena", while "first-person descriptions in musicology draw upon interpretations of intentions attributed to music...moving sonic forms receive the status of actions to which intentionality can be attributed". (Leman 2008) Although Leman's terms organize how music is *described* and most music could be described in both ways, considering when each might be used to describe AI systems is useful. Using a first person description requires a listener to first *perceive* a moving sonic form as an "action", only then can intentionality be attributed to either the human or the technology. To what the intentionality is attributed, then, depends on the performance context. To use Rowe's paradigms, if the intentionality were perceived to be separate from the human's intentionality, creating a "duet", the AI would be a collaborator. If the intentions behind the moving sonic forms are perceived to originate from the human performer, the AI is an assistant, as it would "extend" and "elaborate" the human's intentions (Rowe 1993). Leman's third person description ("repeatable measurements of phenomena") then becomes relevant, as a way of describing interactions between human performers and AI assistants. A measurable repeatability of the AI system's output based on similar human inputs is necessary for the phenomena relating them to be perceived (e.g., every time the human does x, the AI system does y). Furthermore, the repetition of perceiving this phenomena (AI as "extending" and "elaborating" the human's intentions) establishes and maintains it as an assistant (i.e., the perception of shared intention is preserved even as the musical content transforms).

Intention Bonding Another way of considering how systems could be perceived as either assistant or collaborator is by analyzing the *sonic* relationship between the human and technology in real-time performance. In his dissertation, *Laptop Improvisation in a Multi-Dimensional Space* (Pluta 2012), Sam Pluta presents a two-dimensional continuum (temporal offset and sonic distortion, seen in Figure 2) in which to position relationships between human sound inputs and computer music outputs. A computer music system that outputs sound at no (or very little) time delay and no sonic distortion (e.g., just amplification) will easily be heard as relating to and originating from the the human's intention (as an assistant). The further on either continuum (or both) that a system's response is placed, the less likely it is to be perceived as sharing the human's intentions, as it would be less sonically related (more distorted) and/or less synchronized with the human's intentions. Moving along these dimensions can be seen as a process by which the perceived intention of a system is "pulled away" from the intention of the human performer. It is important to see these dimensions (and surely others involved in the process) as continua and not categorical representations. There is, however, likely a distance threshold in this high-dimensional space beyond which the system's output is perceived as separate from the human's intentions. Pluta's continuum clarifies my use of Leman's third person description, which, "provides the results of a measurement". (Leman 2008) The relevant repeatable measurement is identifying where on Pluta's continuum the output of the system lies. Close enough to the origin (graphically and sonically) will prevent first person descriptions, directing one to third person descriptions (likely commenting on the computer music system's placement the continuum, such as, "I repeatedly observe that when the human does x, the computer does y"). Sufficient distance from the origin enables first person descriptions (perhaps about the "duet" nature of the performance). However, perceiving a system's output as separate does not always create perceptions of agency, which is explored further in Section 2.2.1.

Intention in Instrument Design Harry Collins and Trevor Pinch describe the concepts of assistant and collaborator playing out between the desires of synthesizer engineers and musicians saying, "The history of the synthesizer can be seen as a battle ground between the engineers' desire for control and repeatability and the artists' desire for contingency...The engineer wants the machine to be reliable...The artists...want an instrument not a machine–something that will play something unique, something which, although subject to control, is capable of pushing them beyond their own preconceptions–something that

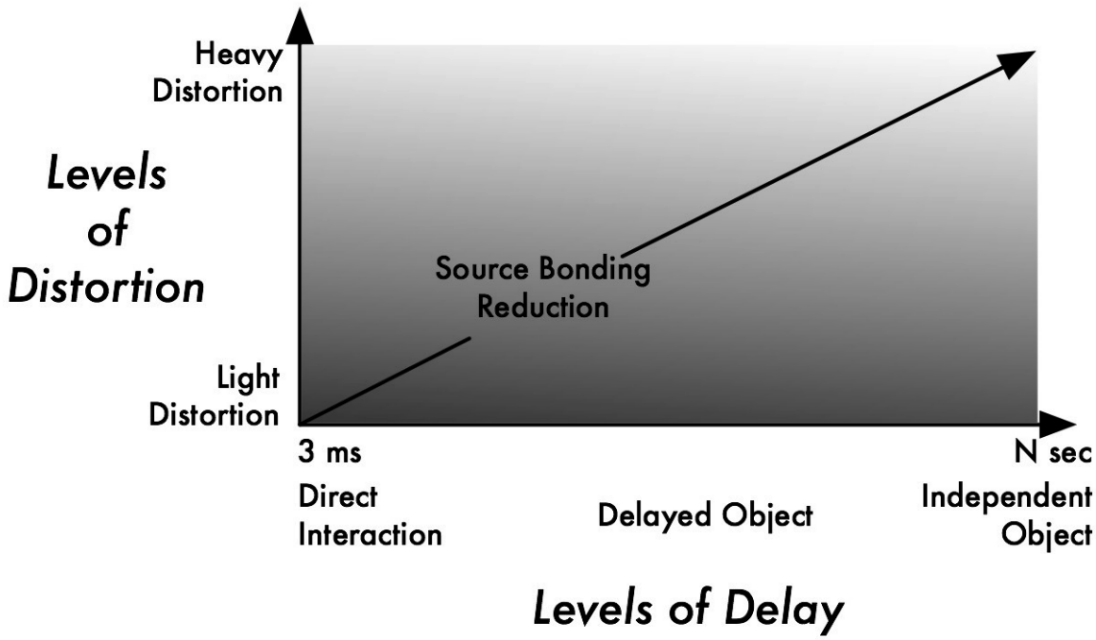


Figure 2: Two-dimensional source bonding continuum taken from Sam Pluta's *Laptop Improvisation in a Multi-Dimensional Space*

can surprise them." (Collins and Pinch 2006) By comparing the desires of synthesizer engineers and artists, they suggest that, in this case, the engineers were approaching the synthesizer as an *assistant* (or a "machine"), while the artists were hoping for it to be a *collaborator* (designated by the label, "instrument", the opposite of Rowe's definition of the term). The engineers wanted the synthesizers to be predictable, preventing any unintentional sounds or effects, ensuring the only perceived intentions would be from the user (thereby creating a "solo" performance experience). The artists wanted to be "surprised" and "pushed"–or more generally be affected by an agent (i.e., collaborator) when working with the synthesizer. The writers also acknowledge the artists' desire for *some* control–some shared commonality or language with their synthesizer/collaborator to then work from. The historical framing of this conflict between engineers and artists anticipates the trend in electronic music of many composers engaging more deeply in the "engineering" side of the practice, perhaps as a strategy to achieve more specific *control* over the *contingency* in their instruments (as will be seen later in the work of Laetitia Sonami). The trend of composer-technician also extends into the current developments in AI for music making, evidenced by the, often large amount of, technical machine learning knowledge needed to engage with machine learning tools for music making.

Compositional Intentionality and Contingency Gil Weinberg makes a delineation between control and contingency also by way of historical comparison: structure centered networks and process centered networks.

> *This differentiation can be related to the tension that emerged in the midst of the 20th century between the radicalization of musical structure and composer control, practiced mainly by 'avant-garde' and 'post-serialist' composers such as Karlheinz Stockhausen and Pierre Boulez on one hand, and the escape from structure toward 'process music' as was explored mostly by American experimentalists such as John Cage and Steve Reich..In such procedural process-based music, the composer sacrifices certain aspects of direct control to create an evolving context by allowing rules (in closed systems) and performers (in open ones) to determine and shape the nature of the music.* (Weinberg 2005)

Here again, *control* is contrasted with concepts like "emergent" and "evolving" where contingency plays a collaborative role in determining and shaping the performance. However, because agency, or "the agent", is harder to identify in Weinberg's process centered networks, it clarifies the role of technology in creating a perception of agency. Works that collaborate with contingency to determine their outcome (i.e., process centered networks), but do not center technology, are less prone to elicit identifications of their contingencies as agential collaborators. This may be because humans are primed to attribute agency to technology or because technology systems are more complex (than say, dice), it is the complexity of the contingency source that enables perceptions of agency.

Meta-Intentionality In discussion of his computer program *Voyager*, George Lewis equates unpredictability with agency, but on multiple time scales, saying,

> *If the computer is not treated as a musical instrument [(to use Rowe's definition, i.e., "assistant")], but as an independent improvisor [(i.e., collaborator)], the difference is partly grounded in the form of program responses that are not necessarily predictable on the basis of outside input...*Voyager's *response to input has several modes, from complete communion to utter indifference...while tendencies over a long period of time exhibit consistency, moment-to-moment choices can shift unpredictably. It is a fact, however, that the system is designed to avoid the kind of uniformity where the same kind of input routinely leads to the same result.* (Lewis 2000)

Lewis' description shows that he views *Voyager* as a collaborator with which he performs a duet, because the program's responses, "are not necessarily predictable on the bases of outside input" (i.e., it does its own thing). The way he has designed *Voyager*, however, adds another layer of agency to the program by creating responses to outside input that range from "utter indifference", showing separate intentions, to "complete communion" (Lewis 2000), in which *Voyager* closely mimics the human's performance, recreating or representing the human's intentions as an assistant (i.e., an "elaborated output exceeding normal instrument response" [Rowe 1993]). This added control over the contingency of the system allows *Voyager* to move between assistant and collaborator throughout one performance giving it an added layer of agency–the ability to change the degree of its agency.

Conclusion To sum up, these accounts reveal how other researchers have identified technology as perceived, described, and designed to be an assistant or collaborator. These writers also acknowledge the fluidity of these distinctions and the precarity of the relation between them. Weinberg mentions the interplay and potential conflict they pose in the creative process saying, "It is important to note that although most networks combine process and structure-based elements, creating a successful balance between these aspects is not a trivial task, as many of the elements are contradictory in nature" (Weinberg 2005). Collins and Pinch hedge against their contingency and control dichotomy saying, "Life is compromise and artists sometimes long for control just as scientists dream of the serendipitous discovery." (Collins and Pinch 2006) At this point an important distinction must be considered: perceiving a technology as *separate* from a human is different from perceiving it as having *agency*. If a system is perceived as *separate*, what is necessary for it to be perceived as having *agency*–as a true collaborator, and not just an additional, *separate* technological element in the performance? Furthermore, how can a human designing a system come to be "surprised" by agency in a system they themselves are designing? How might machine learning as a source of contingency be different (and preferable to?) randomness or analog glitch?

2 Recognizing the Collaborator: Emergent Agency in Complex Systems

The following section sets aside the more simple concept of AI *assistant* to analyze the role of AI *collaborator* in music technology systems. As seen in section 1.1, a system is understood as a collaborator when it is perceived as separate from a human performer and creates a perception of agency. The following section analyzes how this perception arises by first clarifying an understanding of "AI system" in this context, then describing an example in which the perception of agency arises.

2.1 What do we mean when we say Artificial Intelligence?: Towards a Phenomenological Definition

In a recent panel titled "Good Old Fashioned Artificial Intelligence" at the 2020 inSonic conference, the topic quickly turned to the question of what is meant by "artificial intelligence", particularly in the context of creative music making. George Lewis posed the problem well, saying, "It's funny because this whole thing about intelligence, you know, trying to define it is going to be a problem, so you find that defining artificial intelligence is just as problematic and difficult as defining any other kind of intelligence. I mean, there have been many definitions, none of them completely satisfactory". (Karlsruhe 2020) The moderator, Lutger Brümmer, frames the question for the panel as a consideration of whether mathematical models producing complex systems is considered AI, asking, "We forget the context of early AI, it was something like cybernetics, markov chains, different ways how to use randomness, and there were rule based systems,...or there were systems like cellular automata, Lindenmayer systems, fractal systems, and those were, we would today see as part of the generation of complex behavior...So there is something with these mathematical models which create at least complex behavior, now it's a question of if that is–if we could consider this as artificial intelligence or if we should say it's something else." (Karlsruhe 2020) Rather than outlining technological categorizations, the panelists' responses about identifying AI all focus on the perception of intention or agency in collaborative technology. Lewis described that, while performing with *Voyager* "people have to feel that they can get the machine's attention, that they can dialog with it, that it, quote, 'understands' them somehow, and if that's AI, I'm prepared to go with that as one potential method of thinking about AI, I mean it's not the lisp, prologue, scheme kind of AI, but it is a kind of AI." (Karlsruhe 2020) Palle Dahlstadt adds, "I really think...it depends on what role you give to it, if there's a certain kind of complexity–and that threshold is really quite low–it can be perceived as an agent that actually plays with you." (Karlsruhe 2020) He goes on to explain how low this threshold can be, stating that, "even such relatively simple systems that contain complex internal states and latency and feedback, they start to behave like if they, it's like playing with another musician because it's so complex, and that took me a while to realize that these instruments have the same role as when I actually involve much more complex algorithms". (Karlsruhe 2020)

These descriptions demonstrate that for practitioners who have been working with AI in musical creativity for decades, the identification of what is an "intelligent" system is not determined by a category of algorithm or even a degree of complexity, but rather by a perception or attribution of agency in a technological system. The panel also reveals that a common strategy for inducing this perception is the use of a complex system, that of which may involve a machine learning algorithm. This view is nicely summarized by Lewis at the end of his article about *Voyager* titled *Too Many Notes*, saying, "Rather than asking if computers can be creative and intelligent–those qualities, again, that we seek in our mates, or at least in a good blind date–*Voyager* asks us where our own creativity and intelligence might lie–not 'How do we create intelligence?' but 'How do we find it?"' (Lewis 2000) Lewis' suggestion places the onus for identifying artificial intelligence, not in the contents of the machine, but in the perception of the user.

2.2 The Pursuit of Contingency, *or*, the Desire for a Collaborator

Collaboration with Humans The creative invigoration that comes from working with contingent systems similarly operates in collaborative group improvisation. Humans (i.e., living intelligent collaborative systems) are capable of responding to other humans in ways that can act as both collaborators and assistants. Although the perception of intention and agency is seemingly less ambiguous, a human could respond to another in ways that "extend" and/or "elaborate" the other's intentions, acting as an assistant. The fact that a human could produce the same results as a AI assistant, and yet one would not perceive that performance as a "solo", reinforces the distinction that the perception of something or someone as a collaborator is not simply predicated on the difference of the sounds produced but by the perceiver's *attribution* of agency to the "system" producing the sounds. Humans will be more likely to attribute agency to humans than to technologies.

More often, humans act as *collaborators* creating moving sonic forms exhibiting intentions distinct from those of their human collaborators. In their ethnography of improvisers from a human-computer interaction (HCI) perspective, Kang, Jackson, and Sengers describe the dynamic of improvisational contingency, saying, "Several interviewees described the 'tension' stemming from the uncertainty of improvisational process – the ever-present threat of unwanted dissonance and breakdown – as a source of both fragility and potential failure, but also energy and creativity." (Kang, Jackson, and Sengers 2018) As will be seen by examples in Section 3, I have often employed AI collaborators alongside improvising human collaborators as a strategy for increasing the creative energy obtained from contingency.

Kang, Jackson, and Sengers describe the pursuit of this contingency using "coder" terminology, saying, "musicians and artists may seek to exploit or create uncertainty as a mechanism of discovery and expression, making breakdown in effect a *'feature'* rather than a *'bug'*." (Kang, Jackson, and Sengers 2018, emphasis mine) The creative energy derived from the "fragility and potential failure" in group improvisation can be similarly exploited in electronic musicians' use of fragile and potentially failing technology. When describing her improvisational practice, New Renaissance Artist The Honourable Elizabeth A. Baker explains,

> *the thing that I focus on in my improvisation when I'm performing, [is] having something that can subvert my improvisation and cause me to think in a new way. So on tour, I always have one piece of gear that will probably blow up or I make a new configuration at the beginning of a show that will probably go wrong and I specifically do this so that when it goes wrong and I'm shocked out of whatever thing I normally would be doing, I have to think on my feet, and I also have to think on my feet in front of a bunch of humans, so it means that I don't have the luxury of saying, "Oh yeah Elizabeth, this is what you do when this problem goes wrong." It's like, "Oh! This thing is happening. People are watching me. Make it look like you know what you're doing, but now I can't use this thing the way I wanted to use this thing and this has caused all these other problems over here, so what am I going to do now?" So that's my current improvisational take and it's more based on cognitive behavior and what can I do?...I always try to say to myself, "Ok, well what piece of gear in my studio is the least reliable right now?" What can I throw in here that is going to possibly cause a–sometimes it doesn't cause a problem, but a lot of times it does and I find the joy of de-escalating the bomb.* (Baker 2020)

Collaboration with non-Humans For electronic musicians, the desirable moments of "surprise" while using technology (which then allows for those creative suggestions to be pursued or not) has often been rooted in the imprecision of analog systems. Harry Collins and Trevor Pinch write of early analog synthesizers, that they,

> *used transistors and were very imprecise. They were notoriously difficult to control. Musicians would talk about getting an incredible sound at night in the studio only to return to the instrument the next morning to find they couldn't reproduce it. This imprecision was a source of constant delight for some musicians. Famously the legendary space jazz performer Sun Ra took one of the Moog's first synthesizers and he broke every module on the instrument, but the sound it made was "fabulous." The instrument worked better "broken"!* (Collins and Pinch 2006)

In their example, Sun Ra gladly breaks the synthesizer to make it even *less* predictable, and therefore a more desirable instrument. The desire for unpredictability, or systems "difficult to control", that Sun Ra exhibits can also be seen in contemporary artists working with artificial intelligence. Laetitia Sonami–a sound artist, performer, and researcher based in Oakland, CA–says of her work with machine learning, "...in a way, you don't want the instrument to perform like a well-trained circus, you kind of want it to be a little wild, and you want to adapt to it somehow, like riding a bull...I think the machine learning allowed more of this". (CeReNeM 2019)

Rebecca Fiebrink, the creator of Wekinator, gives a compelling analysis of how a machine learning system, as opposed to heuristic approaches, changes the nature of these "bugs" a creator encounters, transforming them from problems into creative opportunities.

> *One of the great things about using machine learning as opposed to coding, as I mentioned it's faster, but also the kinds of mistakes it makes are different. If you make a mistake while you're writing code, often you're going to get a compiler error or...silence, or you get, you know, a filter blowing up. With machine learning the way that a lot of these systems are configured, if it gives you something unexpected it's often still going to be in that parameter space and it might be a sound that you've never heard before, might be a relationship between your input and output that you'd never thought of, but it's going to do something and that's often just more creatively useful than having nothing happen and that can lead you to experiment further.* (CeReNeM 2019)

Fiebrinks's description makes a strong argument for why a creator interested in employing contingent technologies as a collaborator would want to use machine learning: the exercise of trial and error (central to creative practice) is primed for creative surprise–when using AI systems, the "error" is designed to offer new ideas (rather than a crash).

Furthermore, in an interesting turn on the control vs. contingency paradigm of Collins and Pinch, Laetitia Sonami describes using machine learning to *control* the *degree* of *contingency* in her instrument.

> *The unpredictability...depends on how "wide" the machine learning is. If I feed the system training examples whose sounds encompass wide changes based on how I touch the springs, the trained models will move through all these points in unpredictable ways as the springs settle to a resting place. If I give it training examples with narrower changes, the sound will just oscillate slightly as I move the springs. I can thus easily scale the instrument between predictable and unpredictable results by changing how I train. I refer to these variations as the "synthesis terrain"…This "predictability index" is very easily modified and unique to ML.* (Fiebrink and Sonami 2020)

Even with this control over contingency, Sonami still chooses to inject the imprecision of analog mechanisms into her instrument, adding, "I was looking for more complex inputs and opted for a partially chaotic system which would 'fight' the intention of ML and not learn (!). I ultimately used thin springs attached to audio pickups. These would allow for movement of the springs to continue after having been activated by my hands." (Fiebrink and Sonami 2020)

The advantages of contingency in the creative process are recognized by these artists from different angles, including in the use of machine learning. The next section explores how a complex system comes to be perceived as agential by an artist, thereby allowing it to be engaged with as a collaborator.

2.2.1 Emergence of Agency in Complex Systems, *or*, Recognizing the Collaborator

Perceiving Agency... In order to examine the emergence of agency from a complex system, I will compare international sound artist Richard Devine's use of modular synthesizer systems with Deniz Peters experiments using motion tracking of dancers to control sound (Peters 2013). Richard Devine describes using modular synthesizer systems as,

> *Things would happen unexpectedly. It was almost like it had its own personality. It was like a living organism that sort of does its own thing. These circuits would come to life. Even the little slightest thing would cause it to change and be different. I don't know how to explain it. I call it the analog voodoo effect, you know? To a lot of people it's kind of hard to explain unless you have experience with an analog synthesizer. You get this feeling that it has this kind of like, it's almost like it has its own personality to it.* (I Dream of Wires 2013)

By saying that his modular system "has its own personality," "It was like a living organism," and other anthropomorphic phrases, Devine is expressing his perception of intentionality and agency in his modular system. The event that seems to spark this perception is when "Things would happen unexpectedly." This moment of "surprise", similarly expressed by other artists, is also described by Deniz Peters in his study using motion tracking with dancers (Peters 2013). When the dancers would experience "motion tracking glitches", "a foreign agency would seem to gain presence" (Peters 2013). Peters says, "the instrument [(the motion tracking software)] may...turn into [an] agency, particularly if its response is less predictable than that of a static object" (Peters 2013, 159). Unpredictability again appears as a catalyst for perceiving agency.

...as Glitch? or *Goal* Similar to Peters, a lack of predictability is what gives rise to Devine's perception of agency in his modular synthesizer. The difference is that Devine does not describe it as a "glitch". When he describes his experience, it sounds as though he is describing a content generating collaborator:

> *I've been doing a lot of work with analog modular stuff. I work very much the same way in that world too, just kind of start from nothing and then patch up and see what happens, and kind of create this little environment that can generate, sometimes generate things for you that are unexpected. It may give you something that you were looking for, it may give you something that you weren't even looking for that's even cooler than what you were trying to come up with, or you know it might be something that's completely useless. You just never know and that's what I love about it you know it's just kind of throwing the dice out and seeing what happens.* (SweetwaterSound 2014)

Devine's description makes the moment of "surprise" a *goal* (i.e., desirable and/or planned) rather than a "glitch" (i.e., undesirable and/or unplanned). This comparison of *goal* and "glitch" reflects Fiebrink's description of how "bugs" (i.e., glitches) are different when using AI, recasting them as *goals* to be pursued for their creative potential. Similarly, the way Devine describes selecting preferred outputs from his synthesizer system collaborator echoes visual artist Klingemann's description of choosing from thousands of images created by his neural network collaborator.

In order to experience a moment of surprise from a system (*goal* or "glitch"), one must have an established facility with it, such that one's interactions can create predictable outcomes. When one's interactions then produce an unpredictable outcome, one can be surprised. Peters describes this necessary facility saying, "the instrumental action becomes transparent, disappearing as a resistance to our sonic intentions. The instrument seems to become part of one's body. This transparency is a facet of the technical mastery attributed to virtuosity" (Peters 2013). Devine's prolific use of modular synthesizer systems establishes him as a virtuosic user, to whom we can ascribe this facility. Peters' description of the technology as being "part of one's body" establishes it as being an assistant to the performer, as it is synchronized with and extending or elaborating the dancers' intentions (the intended use) and therefore only when it "glitches" does it take on a sense of agency. Here one observes the different descriptions of the "surprise" moments corresponding to different intended uses of the technology. When an *assistant* does something unexpected, it is a glitch, while *collaborators* are intended to do something unexpected.

2.2.2 Separation of the System from the User

During the moment of surprise for Devine and the dancers, the transparency of the instrument and the instrument as an extension of the body, as Peters calls it (Peters 2013), gives way to a realization of the technology as separate from the body and therefore capable of having independent agency. This realization may arrive as a result of the system's output moving away from the origin on Pluta's source bonding continuum. While Peters' dancers experience this surprise via unplanned "glitch", Devine is able to achieve his *goal* of surprise in spite of his virtuosity. I say "in spite of" because one may suppose that a "virtuosic" user performing virtuosic execution would be in control at all times (and therefore not-surprisable), as one might consider the execution by a virtuosic violinist. A question then, is how does Devine, a virtuosic user of these systems, achieve his "goal" of surprising himself? How is a virtuosic user of this instrument able to create a transition from control to contingency, solo to duet, assistant to collaborator, thereby creating an agent with which to collaborate?

2.2.3 The Role of Complexity

The agency of a complex system is an emergent property of its complexity. Agency emerges when the system becomes too complex for the user to keep track of all the interconnections and relations necessary to precisely predict the outputs of the system. Furthermore, creating this agency is a goal of using the system, as one can then employ that agent as a collaborative partner in the creation of music. Once Devine constructs the modular system to be sufficiently complex to perceive agency he can then employ that agent as a creative collaborator.

When beginning to create a modular synthesizer patch, Devine says that he "just kind of start[s] from nothing and then patch[es] up and see[s] what happens" (SweetwaterSound 2014). Early in this process, when there are few interconnections, Devine, a virtuosic user, would be able to provide Leman's third person descriptions of the sound being produced. Given a particular set of connections he would be able to account for the sonic results using objective, measurable descriptions such as, for example, "the square wave oscillator is being filtered by the resonant low pass filter, the cutoff frequency of which is being modulated by a triangle wave LFO." Descriptions such as this are objective enough to recreate this sound in the future or by someone else. Using third person descriptions at this stage points towards the modular system currently being an assistant[2]–it is not offering surprise, only extending or elaborating (i.e., sonifying) the intentions of the user.

As one works with a patch, the complexity of it tends to grow, ultimately to the point where one, including virtuosic users such as Devine, are no longer able to keep track of all the interconnections contributing to the resulting sound. At this level of complexity, one is no longer able to readily provide a third person, objective, reproducible description of the system. Instead, one starts using subjective, first person descriptions that, as Leman says, "draw upon interpretations of intentions attributed to music" (Leman 2008). For example, Devine uses phrases such as, "a living organism that sort of does it's own thing," and "These circuits would come to life" (I Dream of Wires 2013). This is the first step in the process by which agency emerges from complexity in modular synthesizer systems. Devine achieves his *goal* by pursuing complexity to the point where he is surprised by the sonic results, indicated by the use of first person descriptions.

It is important to note that although a user like Devine becomes unable to keep track of all the interconnections of a system, he does not loose track of the system itself. One is still able to understand what the system is doing (e.g., making frequency modulation sounds) and how it is doing it (passing around electrons), but is unable to predict other parameters such as when and which different variations of sound might occur.

[2]For demonstration purposes, describing this instrument as an assistant is useful, however, it doesn't fully agree with my definition of AI assistant in that the task it is performing is not very complex and does not involve data processing.

2.2.4 Mirroring, *or*, Recognizing Oneself in the Collaborator

The second step in the emergence of agency from a complex system is described by Leman, saying, "Attribution of intentionality is likely to occur on the basis of mirroring, that is on the basis of a simulation of the perceived action in the subject's own action. Actions of others are understood as intended actions because the subject can simulate them and understand them as its own intended actions." (Leman 2008) He clarifies that, "This intentionality can be attributed to subjects as well as to objects (or, rather, events)." (Leman 2008) Leman explains that it is not simply enough for one to perceive a system making separate sounds that surprise the user, the perceived intentionality of the system's sounds must also resonate with the users sense of their own intentions. Leman describes the perception of agency as an emergent property, stating, "Through motor resonances, the complexities of the physical world are related to our personal experiences. Intentionality, therefore, can be conceived of as an emerging effect of this communicative resonance." (Leman 2008, 102) Regarding modular synthesizers or AI music systems, in order to perceive the system as an agential collaborator it must be perceived as a separate, but *similar*, actor. The actions (i.e., sounds) it makes when it surprises the user are recognizable as sounds that the user could make and might desire to make using such a system.

2.3 Conclusion

According to Leman, moving sonic forms that are perceived as separate from a human performer, yet exhibit behavior that mirrors one's own intentions can be perceived as agential. These perceptions of agency point toward a phenomenological, rather than categorical or technical, definition of AI collaborators. Electronic musicians have always be interested in employing contingency, such as these agents, as collaborators in their creative practice. This is often achieved by creating technological systems complex enough to create moving sonic forms that are surprising to the the system designer and/or user.

3 Practice-based Research in AI for Music Making

All of the code for these projects can be found in corresponding Appendices.

3.1 Using Audio Descriptors

Much of the research that follows in this section is based on datasets created by audio analysis. In order to create these analyses, the audio is often first sliced into short segments either all consecutive and equal in

Index(es)	Audio Descriptors
0-39	40 MFCCs
40	Spectral Centroid (Hz)
41	Spectral Spread (Hz)
42	Spectral Skewness (normalized) as a Ratio
43	Spectral Kurtosis (normalized) as a Ratio
44	Spectral Rolloff (Hz)
45	Spectral Flatness (dB)
46	Spectral Crest (dB)
47	Frequency (Hz)
48	Frequency Confidence (0-1)
49	Loudness (dB)
50	True Peak (dB)
51	Zero Crossing (Hz)
52	Sensory Dissonance (0-1)
53-92	40 Mel Bands (in amplitude)
93-104	Chromagram (12 TET)

Figure 3: List of all 105 audio descriptors extracted with author's SynthMIRNRT SuperCollider Class

length (usually between 20 - 100 milliseconds) or through analysis, making slices at loudness onsets or novel changes in spectral shape. These slices are then analyzed with a fast fourier transform (FFT) and descriptors are calculated from the spectra (some descriptors such as RMS or zero crossing do not require an FFT). Some of the analysis descriptors used are seen in Figure 3, most of which are extracted using tools from the FluCoMa Project (Tremblay et al. 2019). Some code for this extraction can be seen in Appendix A.

3.2 Artificial Intelligence as Assistant

Returning to the idea of *assistant*, the following two examples are uses of neural networks as machine learning assistants.

3.2.1 Timbral Classification for Sound-to-Light Parameter Mapping

In multimedia performance the synchronization and coordination of different media is often an aesthetic goal. If the multimedia elements (e.g., sound, light, video, etc.) are fixed before the performance (as "fixed media"), the relation between them can be pre-composed, not requiring real-time reactivity.[3] If the components are not fixed (such as in improvisation), then media elements must be manipulated in real-time in some way. In the case of performance with audio and lights, one undesirable strategy would be to have a separate control interface for both sound and lights. In order to create the perception that the

[3] I used this strategy to compose *tap*, for percussion trio, tape, lights, and video: `https://vimeo.com/339268455`

sound and lights are related, the performer would have to manipulate the sound and lighting instruments simultaneously and with similar behavior. A more common strategy is to create real-time "audio-reactive" media elements that change media parameters in response to changes in sound. Real-time audio analysis creates streams of data that can then be mapped onto multimedia parameters such as color, brightness, or event triggers.

My composition *feed* is a structured improvisation for bassoon and multimedia that uses lighting instruments reacting in real-time to sound generated by my improvised no-input mixer performance. My initial attempt at this live reactivity was a matrix-based scaling and summing design[4] that mapped audio analysis descriptors to lighting parameters (such as color and brightness). After two performances with this system in October 2018 and April 2019, I decided that the sound-to-light mapping scheme was not as audio-visually compelling as desired. The visual activity of the lights did not match the clarity of the sonic changes or the distinctness of the sonic spaces created by the no-input mixer because the linearity of the mapping system was not able to strongly portray the non-linear complexity of the no-input mixer's sonic properties through the lights.

The solution I chose to implement was a neural network used to classify, in real-time, which sonic space the no-input mixer was currently sounding. The non-linear property of neural networks (created by, in this case, the sigmoid activation function in the hidden layers) is more capable of representing the complexity of the no-input mixer's sound. I chose to not pursue a heuristics-based classification algorithm because (1) determining the best parameters would be very tedious and time consuming (that is what a neural network does for you!) and (2) if in the future I chose to use a different sound source, the system could easily adapt by simply creating a new training set.

The categorical differentiation provided by this neural network more accurately reflects the experience of hearing the no-input mixer switch between different sonic categories. In order to exhibit these sonic categories clearly with the lights, each category was assigned a different set of audio analysis-to-lighting parameter mappings (such as, an increase in volume creates an increase in brightness). When the neural network identifies a change in sonic space it triggers a visual change—the parameter mapping scheme is switched to the one prescribed by the new sonic space. Figure 4 shows how these system components work together.

In order to create the training data for supervised learning, I recorded audio from all four sonic spaces of the no-input mixer (as performed and identified by me: low thuds, high squeal, distorted noise,

[4]Similar to (Brandtsegg, Saue, and Johansen 2011)

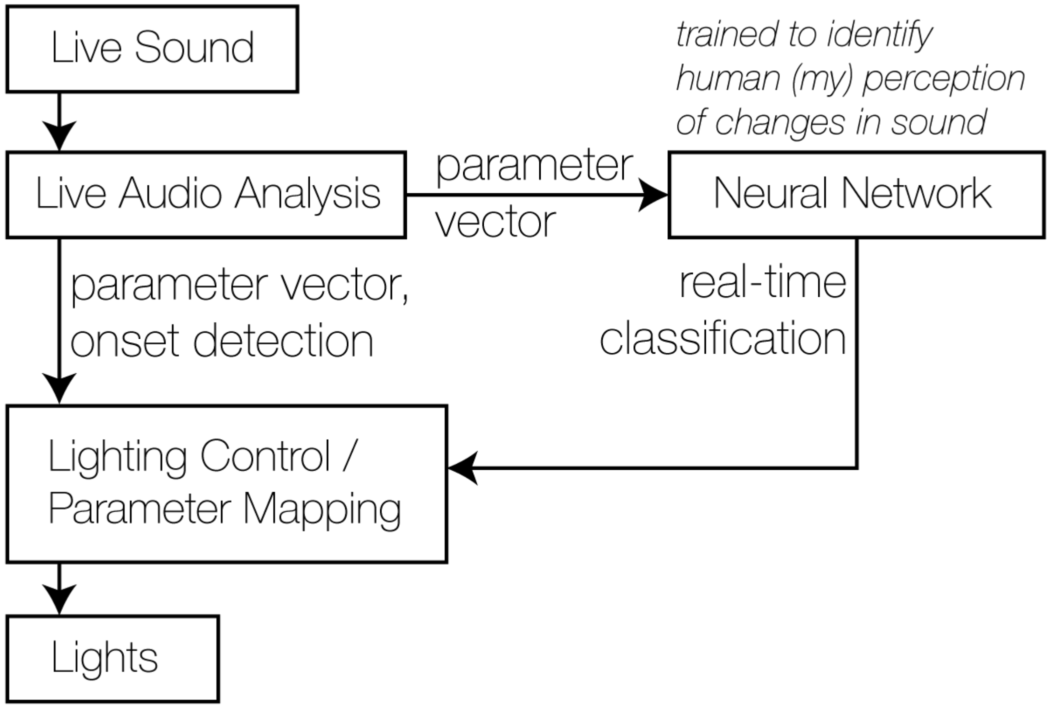


Figure 4: Data flow diagram of sound-to-light mapping system for *feed*.

and quite sustained noise). From each category I extracted audio descriptor vectors derived from consecutive 30 millisecond windows[5] and paired each analysis vector with its one-hot encoded category vector. A one-hot encoding is a vector the length of the number of categories the neural network is learning to identify, with zeros in all places except at the category-identifying index, which is represented by a one. The neural network used was one I created for this composition that trains and predicts on the client side of SuperCollider. The audio-visual correspondence that this new system created was much more compelling. Retaining the strategy of mapping analysis parameters to lighting parameters (from the first iteration of the work), maintains the gestural nature of the lights' audio reactivity, while using the categorization of the neural network to switch between different mappings strengthens the visual correlation of the fast-category-switching sonic experience. I have since employed this system it in multiple performances, such as *shadow*.[6]

[5]This is the same window size to be used in the real-time analysis, which creates an update frequency, or visual frame rate, for the lights of about 33 Hz, which is well above the commonly used framerate between 24 and 30 frames per second for video.

[6]`https://www.tedmooremusic.com/shadow.html`

Using a neural network in this way is a clear example of using artificial intelligence as an *assistant*. To use the undesirable example from above, the performer could play the no-input mixer and each time they identify (or execute) a change in sonic space, they could reach over to the lighting controls and manually switch the mapping scheme. Instead of this strategy, I have chosen to have a neural network *assist* me by identifying changes in sonic space and switching the mapping scheme for me. I am not asking the neural network to perform any creative task or to engage with me as a performative collaborator, but simply to assist with a task—one that I could do myself, but instead choose to focus on my sonic performance. The system is not designed to be complex enough to create "surprise" but instead is intended to behave predictably.

The idea of artificial intelligence as "assistant" corresponds to the paradigm of supervised learning because supervised learning requires that a human labels all the data (in this case categorizes the sounds) before training. The human must already know the "correct" answers that the algorithm is being asked to learn, and once the learning has happened, the human can rely on the algorithm to perform that task for them, similar to how a human supervisor may engage a human assistant.

3.2.2 Frequency Modulation Resynthesis

The "n-to-m" mapping paradigm is a commonly used strategy for synthesis parameter control (Hunt and Wanderley 2002), especially when the number of control parameters are more than one performer is able to manipulate at one time or the synthesis parameters are being controlled by atypical information streams (such as motion detectors (Peters 2013) or photoresistors (Fieldsteel 2018)). Often these mappings are done with matrix-based linear scaling and/or weighted sums (Brandtsegg, Saue, and Johansen 2011). Using a neural network with hidden layers and non-linear activation functions allows for more complex relationships to be established, but also allows for mappings to be learned from training data, rather than chosen by a designer.

For this study, I use a neural network to create an n-to-m mapping from audio analysis descriptors to synthesis parameters for frequency modulation (FM) (Chowning 1973) using regression. FM was chosen because it is a commonly employed synthesis algorithm and is capable of producing a large variety of timbres and morphologies using very few input parameters. The neural network receives audio analysis descriptors as input, from which the network predicts the FM synthesis parameters that most closely recreates the input audio signal spectrum. Because the prediction and synthesis happen nearly instantaneously and the limitations of FM make it very likely not able to precisely reproduce the input sound, the result resembles a real-time distortion effect on the input signal.

The success of this effect depends on adjusting two main hyperparameters: (1) what points to use in the dataset for training and (2) which audio descriptors to use as inputs to the neural network. One testing harness was created for examining both, which was done iteratively by adjusting each hyperparameter separately.

The most challenging decision to make, which also took the longest to solve, was what points to use in the dataset for training. Each data point needs to consist of an output vector of FM synthesis parameters (y) (carrier frequency, modulating frequency, and index of modulation) paired with an input vector of audio descriptors (x) derived from the FM audio signal created by the parameters in the output vector. Supervised training with this dataset allows the neural network to learn to predict what synthesis parameters were used to create a sound (based on that sound's analysis descriptors).

To create the training set, I initially chose to create 30 steps per FM synthesis parameter ($30^3 = 27000$ data points in total). The carrier and modulation frequencies were scaled exponentially from 20 to 20,000 hertz (to match human perception of pitch and cover the range of human hearing) and the index of modulation, which is essentially a coefficient in the algorithm, ranged from 0 to 20 (a commonly used range). Then, using SuperCollider's non-real-time functionality, I iterated over each data point and used these parameters to synthesize a signal from which audio descriptors were extracted. In order to create a fluid workflow for later analysis I chose to extract 105 descriptors (Figure 3) from this analysis, as I can then later examine which work best for training.

My criteria for selecting which audio analysis descriptors would create the best trained neural network were (1) minimizing error with cross validation and, more importantly, (2) a more compelling sonic result. The four analysis vectors I tested were (1) 39 MFCC values[7], (2) 40 Mel Bands, (3) 92 of the 105 analysis parameters (Figure 3, all except the Chromagram), and (4) seven spectral descriptors (centroid, spread, skewness, kurtosis, rolloff, flatness, and crest). Although none of these were yet as successful as I desired, I found that the seven spectral descriptors was the best.

My first attempt to improve performance was to use a larger dataset that was less evenly spaced throughout each parameter. Poisson disc sampling creates a dataset that consists of randomly placed points evenly distributed throughout the entire parameter space (Bridson 2007). Using this algorithm I created a dataset of 37,542 points in normalized space which was then transformed using the same scalers as before to create a complete training set via FM synthesis and audio descriptor analysis. Although this

[7] I used an analysis of 40 MFCC values but ignored the first one as is commonly done, because it essentially represents amplitude.

solution did not improve or worsen results, I continued using this larger dataset for training with the understanding that more data points could yield more accuracy.

After analyzing the properties of my dataset, I realized that the neural network was not converging well because the training data had many points that were likely confusing it. Many of the FM parameter combinations create spectra with negative frequency sidebands that "fold" around 0 Hz back into the positive frequency domain (Dodge and Jerse 1997). Similarly, many of the spectra would create aliasing in the upper register.[8] These phenomena will create spectra (and resulting descriptor vectors (x)) that may be similar to other spectra synthesized from very different FM parameters (y). A neural network that is shown very similar inputs (x) which are paired with diverse outputs (y) may become confused as it tries to regress towards different outputs from a similar input space. In Figures 5 and 6 one can see two examples of my analysis of the training set. Looking at the top graph of each, the more that one color is spread out in the two dimensional space, the harder it may be for the neural network to learn to predict a certain output (location in 2D) from its input (color) because similar inputs (colors) point to multiple output spaces (2D space).[9] In order to fix this problem, I iterated over the dataset, removing any data points that did not meet the following constraints:

1. carrier frequency < 5,000 Hz *(more common range and should minimize aliasing)*
2. modulating frequency < 2,500 Hz *(more common range and should minimize aliasing)*
3. ((index + 1) ∗ modulating frequency) < carrier frequency *(should minimize/eliminate negative frequencies folding around zero)* The heuristic of using the idndex of modulation + 1 is taken from (Dodge and Jerse 1997).

Filtering the data for these criteria left 5,685 points. Even though this dataset had fewer points, the resulting sound was much more convincing. The pitch and timbre of the synthesizer was phenomenologically more similar to the input that previous attempts. The neural network and FM synthesis were able to recreate the pitch, noisiness, and morphologies of the input sound, making it a useful "distortion" type FX of an incoming signal.[10] Trying different neural network architectures revealed that one hidden layer of six neurons created the lowest loss.[11]

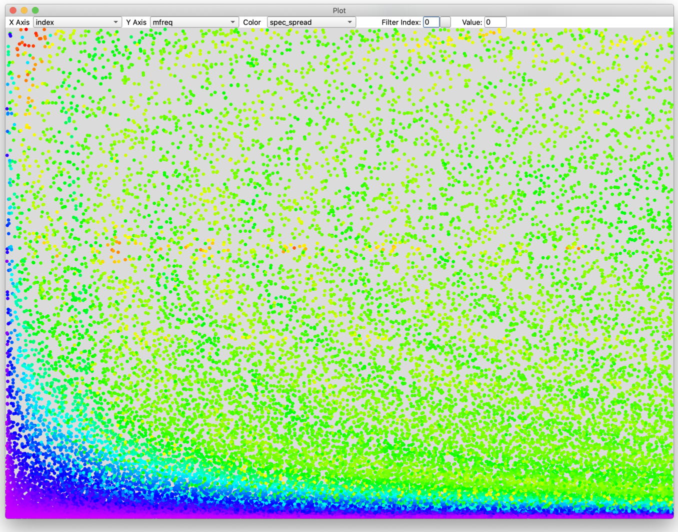
AI and creativity researcher Rebecca Fiebrink explains how data can be used as a control

[8]The synthesis was done at a samplerate of 44.1 kHz.

[9]Although these few dimensions do not fully visualize the problem, these charts are used to visually consider the concept, as a metaphor for the larger conceptual issue being faced.

[10]Some representative results can be heard here: `https://drive.google.com/drive/folders/1H51GyG6eJH5QTXzLXguOI5lmS5H978mj?usp=sharing`

[11]The neural network framework that was used is from the FluCoMa Project. (Tremblay et al. 2019, `https://github.com/flucoma`) Activation function of hidden layers = sigmoid; Activation of output layer = identity; Number of epochs trained = 31,800; Final loss = 0.0499; All input and output data were normalized for training and testing.



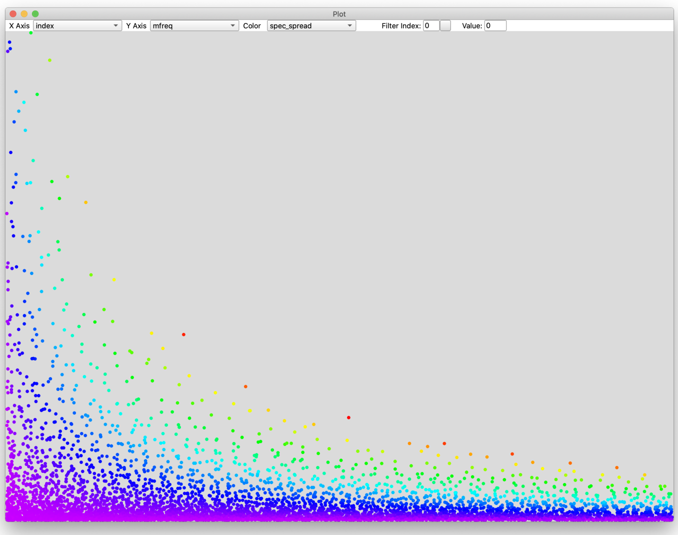


Figure 5: Spectral spread values (color) for each data point represented by its index (x axis) and modulating frequency (y axis). Top graph shows *before* filtering data. Bottom plot shows *after* filtering data.

interface to tell the computer what to do, saying,

> *In most machine learning classes that you might take, we talk about data as ground truth. Data is something you gather from the world and you try to make a really accurate model of that data because probably you don't understand as a human how the stock market works but you want to predict it more accurately or you don't understand how a complicated set of medical test work together and you want to predict more accurately whether some treatment is appropriate for patient. But here we're not using data in that way. Data is actually instead an interface for somebody to communicate to a computer. I'm communicating through these examples I'm giving, what kinds of movements I want to make and what kinds of sounds I want to be paired*

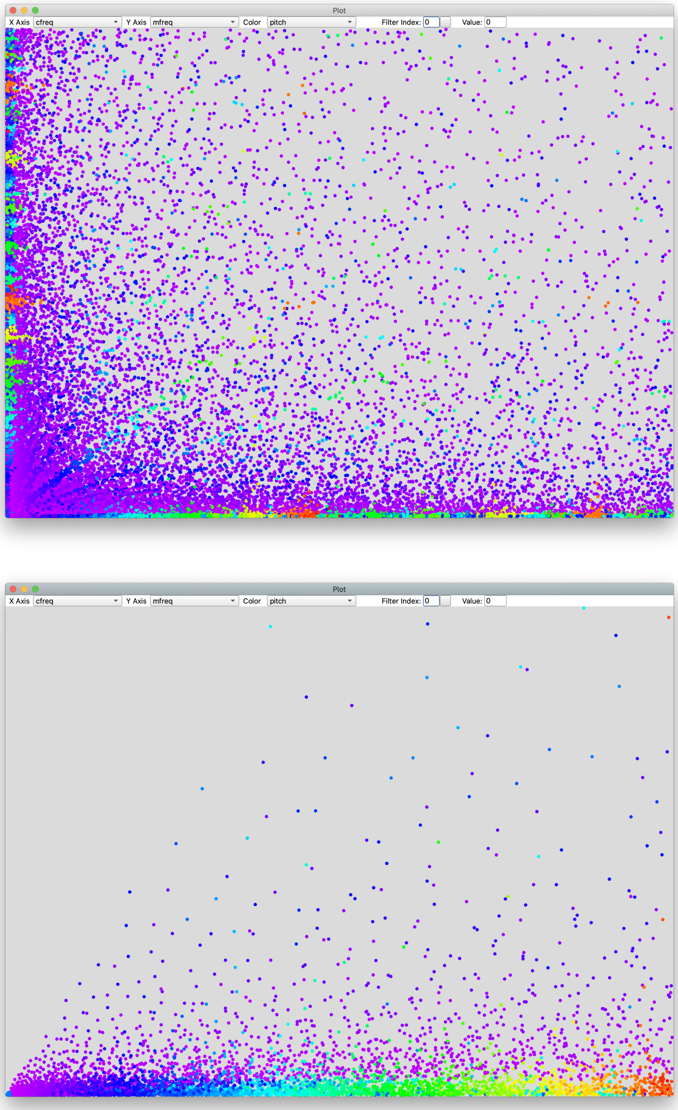


Figure 6: Frequency values (color) for each data point represented by its carrier frequency (x axis) and modulating frequency (y axis). Top graph shows *before* filtering data. Bottom plot shows *after* filtering data.

> *with those movements. It might be fairly arbitrary, or not. It might be subjective, but in either case, I'm, sort of, the expert. I'm using data instead of code to tell the computer what I want it to do.* (CeReNeM 2019)

By not including data points that would create aliasing or negative frequencies folding around zero I am using my dataset to make clear to the neural network that I do not want it to create FM parameters that would create aliasing or negative frequencies folding around zero.

3.3 Artificial Intelligence as Collaborator

3.3.1 The harmonic series strikes again: emergent tonality in feedback resonant tubes

My composition *hollow*, includes large PVC tubes that are used in a complex audio feedback system. The three tubes (all four inches in diameter) are cut to lengths of 10 feet, 8.4 feet, and 7.5 feet in order to achieve resonances at 55.8 Hz ($\approx$ A1), 66.3 Hz ($\approx$ C2), and 74.4 Hz ($\approx$ D2) respectively. Each can be controlled through digital processes to direct the system's resonance toward particular frequencies or, alternatively, allowed to behave autonomously, being regulated by negative feedback mechanisms in the feedback signal path. In each case, the tubes act as filters, creating resonance at frequencies in a harmonic series, the fundamental of which is based on the length of the tube. As a whole system, the three tubes can be operate in parallel as three different feedback systems, or in series, creating one large feedback system that circulates through all of them.

Building up the Complexity After initially discovering with one tube the beautiful tones that sound when a feedback loop is created, I then chose to add two more tubes of different lengths to create a richer harmonic palate. Once the three feedback systems were sounding, a clear next experiment was to hear the tubes in series, which, by increasing the complexity of the system, provided some surprising results including a distinct A Mixolydian-type scale. Continuing to experiment with this instrument and adding feedback saxophone in performance increased the complexity of the system, which I began to perceive as an agent, offering collaborative and creative input to the rehearsals and performance. The performative and musical content that this collaborator offers are the sequence of tones, harmonies, timbres, and gestures that are created as the tubes interact with each other while in series. The goal of the following analysis is to understand the emergent properties of this feedback system so that they may be further exploited and/or so that future feedback system designs can begin from creative goals based on system criteria or heuristics.

Tube Controller Each tube has a microphone at one end and a speaker at the other (both placed directly in front of and facing their respective openings). When the tubes are operating in parallel, the sound that comes out of the speaker travels down the tube, is picked up by the microphone, amplified, and sent back to the speaker, creating a feedback loop. Between the microphone and speaker, this signal goes through SuperCollider to be processed by two compressors, a limiter, a tanh transfer function, and a softclip transfer function to keep it from clipping or distorting unpleasantly. SuperCollider also enables the performer two ways of manipulating the audio in the feedback path: Partial Mode and Modulation Mode

(Figure 7).

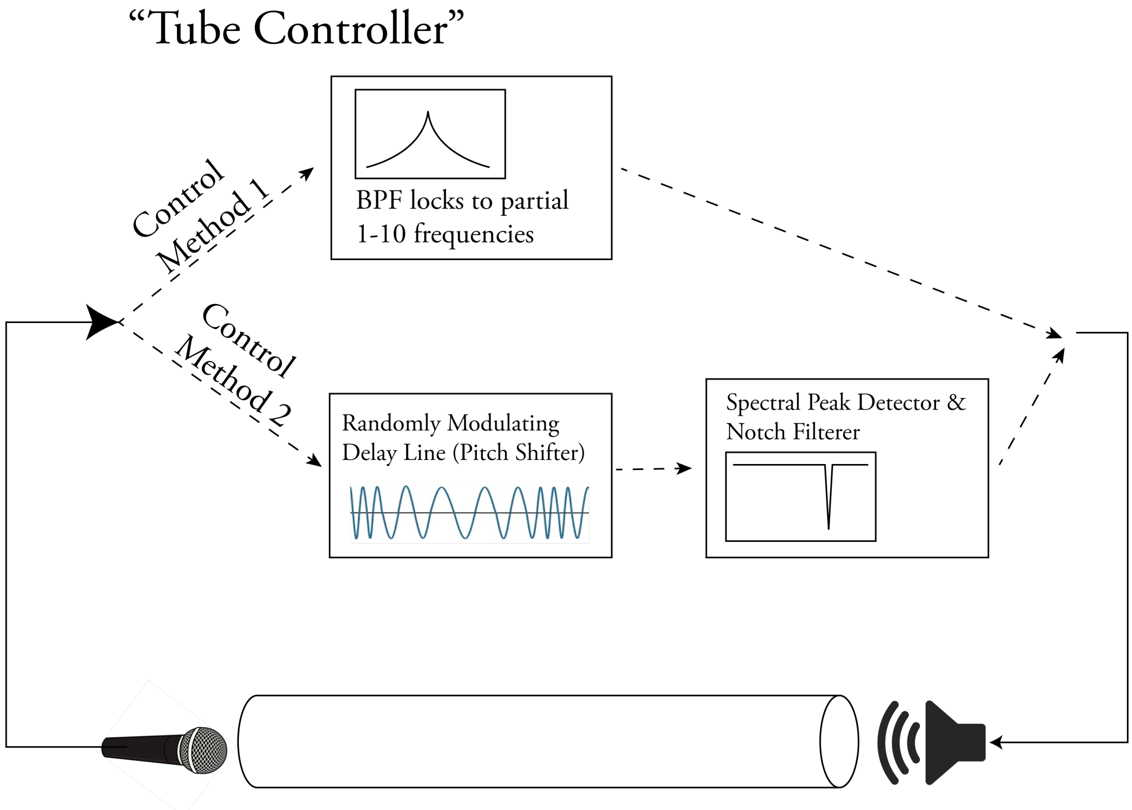


Figure 7: Feedback tube signal flow including control methods 1: Partial Mode & 2: Modulation Mode.

Partial Mode The first method of manipulation is with a bandpass filter, the center frequency of which is only able to be positioned at resonant frequencies of the tube it is controlling (partials 1-10). The performer can freely move the center frequency along a slider that "snaps" to these limited options (Figure 8). The rejection of other frequencies (and resonance of the filter) restricts the feedback from sounding anywhere other than at the partial indicated, however, presence of the tube and multiple transductions involved introduce some analog imprecision into the system, preventing it from resonating at precise integer multiples of its fundamental. Experientially, however, the sound of an overtone series is still very strong. Figure 9[12] shows a histogram of each tubes' sounding frequencies (while the tubes are operating in parallel). Dotted lines indicate partials for each tube by color. Throughout this analysis the tubes are represented by the following colors: A fundamental: red, C fundamental: green, D fundamental: blue.

[12]The following analyses are created from pitch tracking information analyzed from each tube's microphone.

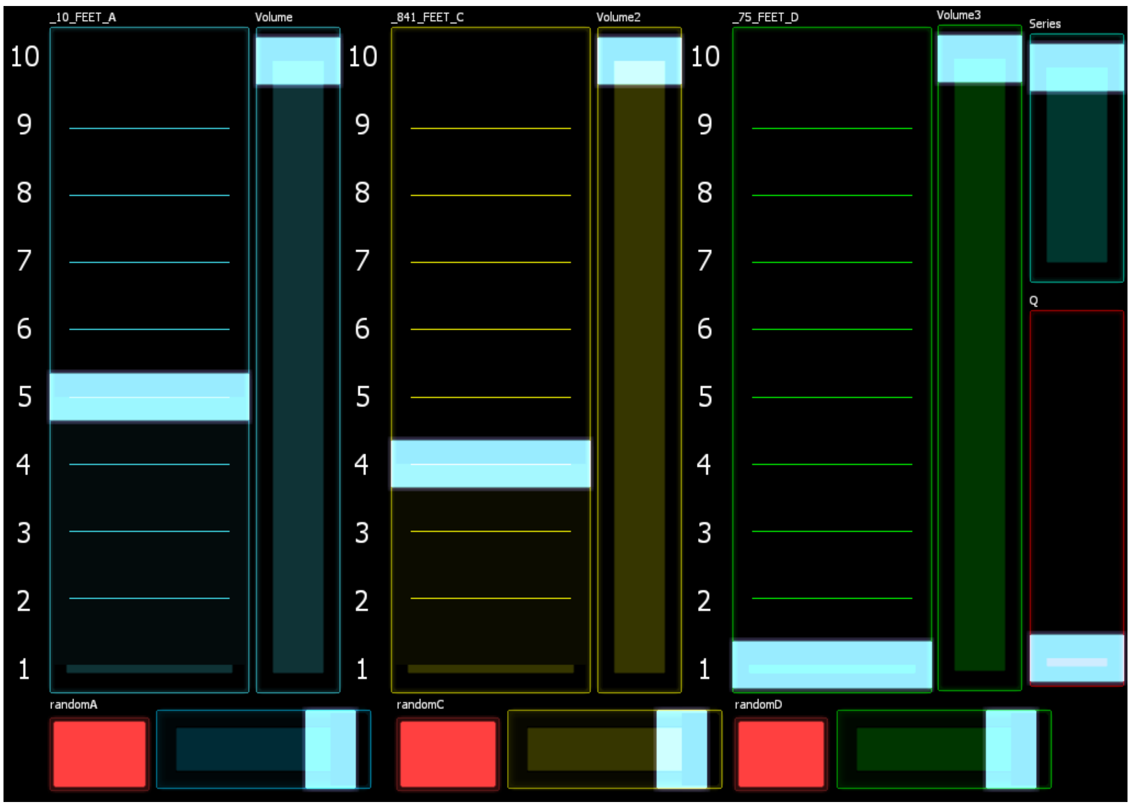


Figure 8: Screenshot of "tube controller" interface for *hollow* created in the iPad app Lemur.

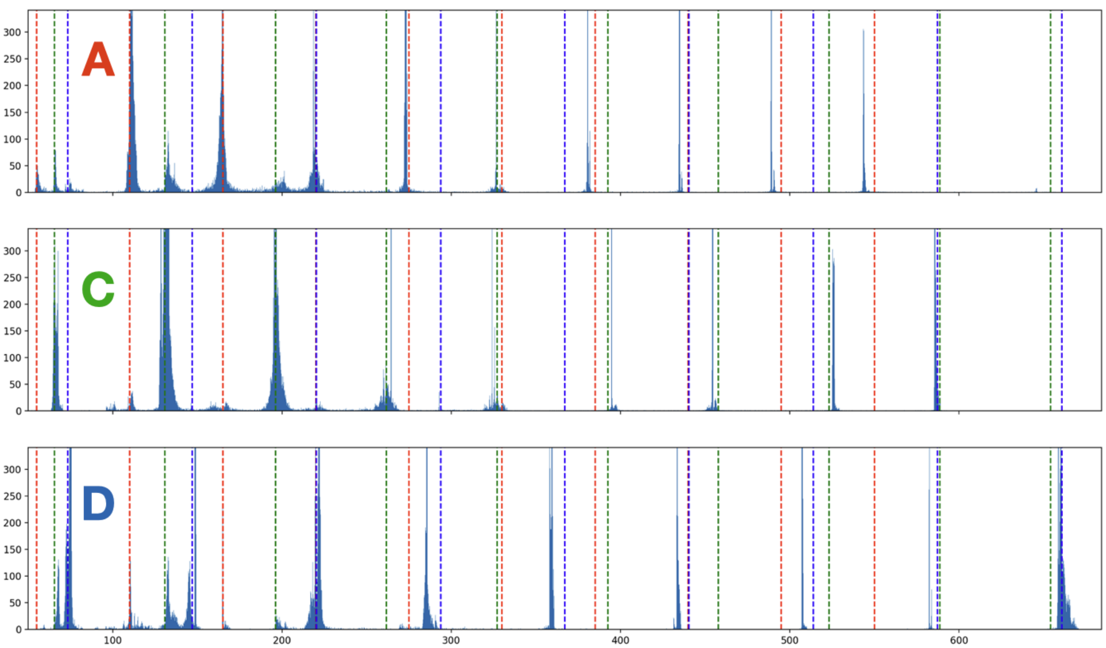


Figure 9: Histograms of tubes' sounding frequencies when operating independently.

Modulation Mode The second method of manipulating a tube's feedback audio is with a combination of a modulating delay line and a spectral resonance suppressor, which allows the tube to behave more autonomously. The modulating delay line (sine wave low-frequency oscillator at 0.01 Hz with a depth of 0.06 seconds[13]) acts as a pitch shifter, "pushing" the signal way from its current resonance. The spectral resonance suppressor uses an FFT analysis[14] to identify spectral peaks surpassing a given threshold and responds by attenuating the peak band with a narrow bell EQ (q = 20) that fades from 0 to -2 dB over 4 seconds. Although -2 dB seems minimal, it continuously adds these bell EQs until the spectral peak is below the threshold. Also, it adds the bell EQ at whatever frequency the phase vocoder is currently reporting for the bin with the maximum magnitude, therefore even if the peak shifts in frequency but stays within the same bin the suppressor will track it. Each bell EQ that is added stays in place for a random length between 14 and 17 seconds. This negative feedback system counteracts the positive feedback of audio amplification, keeping the system from continuously growing in volume, but also preventing it from resting on one resonant frequency for very long.

Performance In the opening of *hollow*, the tubes are in parallel, used in Partial Mode, creating three independent voices used to create three part harmonies with rich beating patterns (Figure 10). After developing these sounds from their lowest to highest register, I begin switching the tubes, one by one, into Modulation Mode allowing the modulated delay line and resonance suppressor to act on the signal, preventing it from remaining at any one partial for long. After letting these overtone series sound for a while, I cross fade to the signal flow that connects the tubes in series (seen in Figure 11), now creating one feedback loop, instead of three. As expected, histograms of the tubes' analysis frequencies while in series (Figure 12) are more similar to each other showing clear preferences for where, in frequency space, the three-tube feedback system prefers to resonate. The sonic experience of these tubes while in series is a slowly evolving soundscape that draws from tones in an A Mixolydian scale with an added C natural in the lower register. Although only one frequency is most prevalent at any given moment, other pitches from this scale can be heard at various times creating a sense of harmony, especially during moments of transition from one salient frequency to the next.

[13]Upon reflection, I am not sure this delay line would have much of an effect. I do believe that the slight pitch shifting may slow down the rapid increase in volume made by the system's resonance. Unfortunately the tubes are inaccessible for further testing.

[14]4096 samples; hop size = 1024 samples; window = hanning

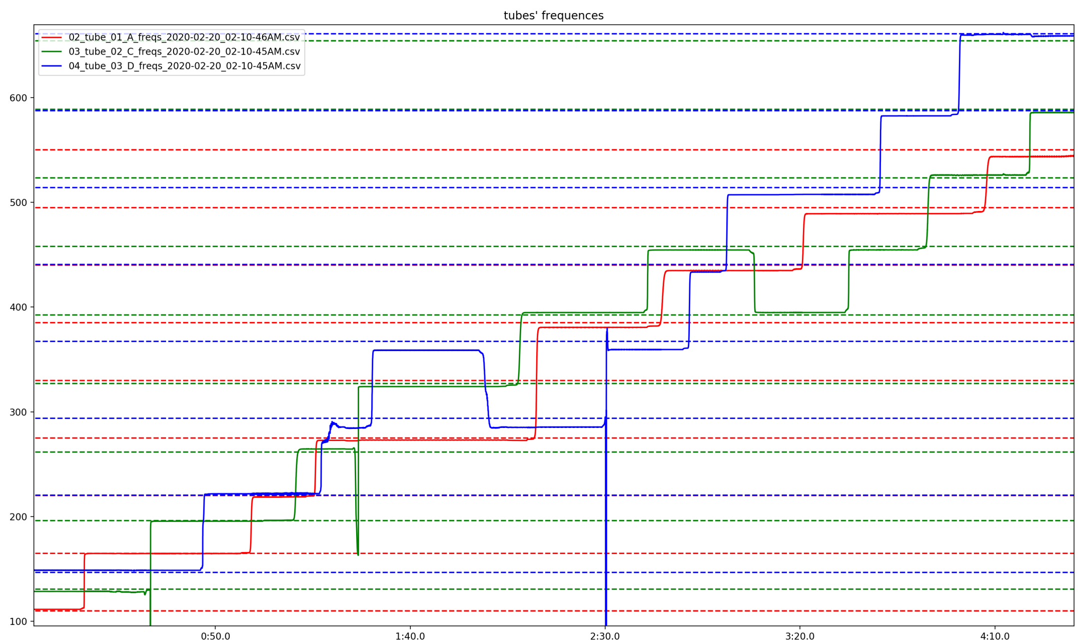


Figure 10: Plot of tube frequencies for opening of *hollow*.

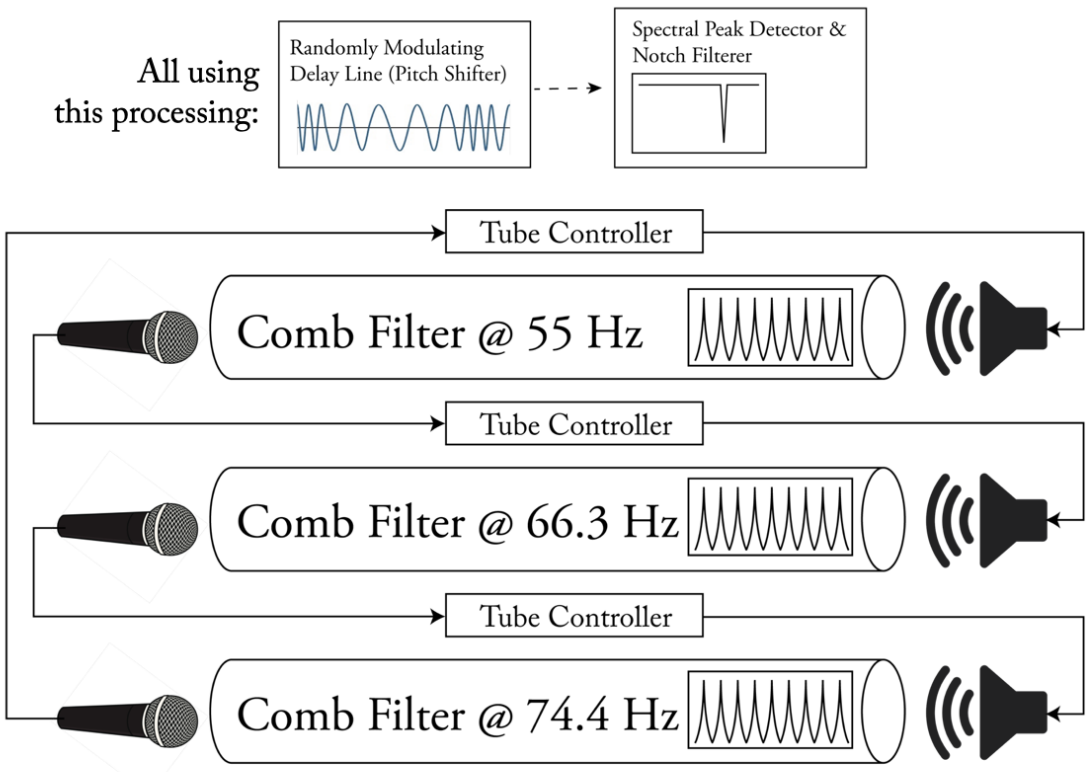


Figure 11: Signal flow of tubes in series.

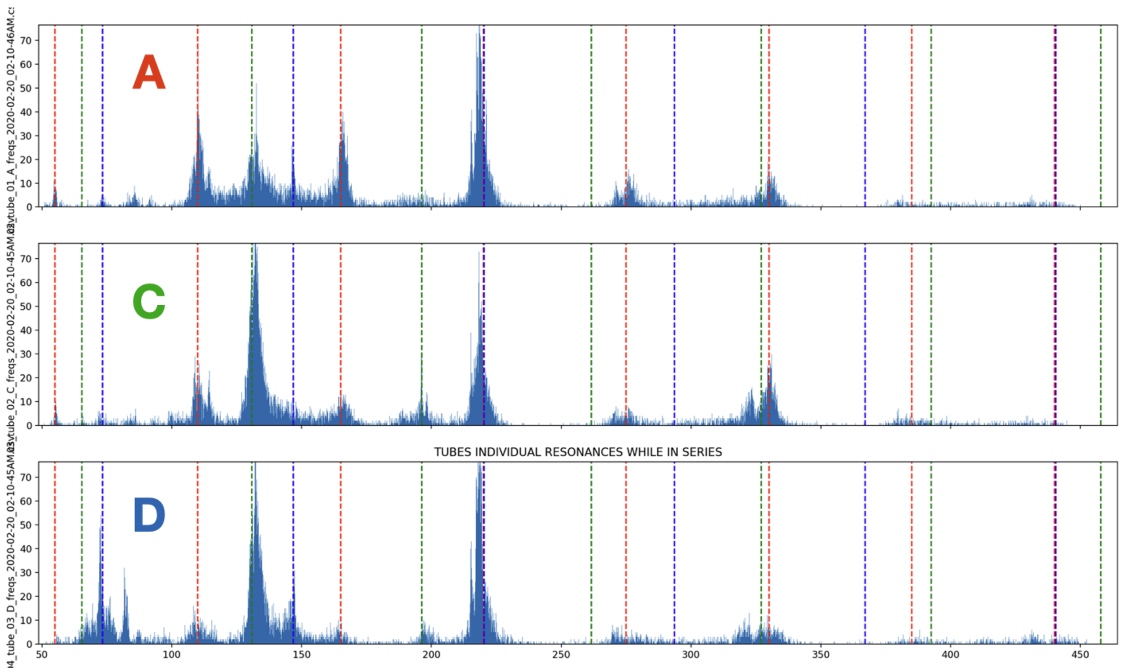


Figure 12: Histograms of the tubes' analysis frequencies while in series.

Frequency Cycling Charting the three tubes' frequencies though time (as seen in Figure 13) reveals them moving mostly in concert with each other, as well as a clear periodicity in the occurrence of certain frequencies. While this first seems like an indication of emergent behavior resulting from complex interactions, I quickly realized that it is most likely caused by the chosen duration of the bell EQs in the resonance suppressor. The cycle of frequencies seen in Figure 13 is about 20 seconds: just longer than the range of each bell EQ (randomly between 14-17 seconds), accounting for a few seconds for the feedback to build up in a register after the EQs are removed.

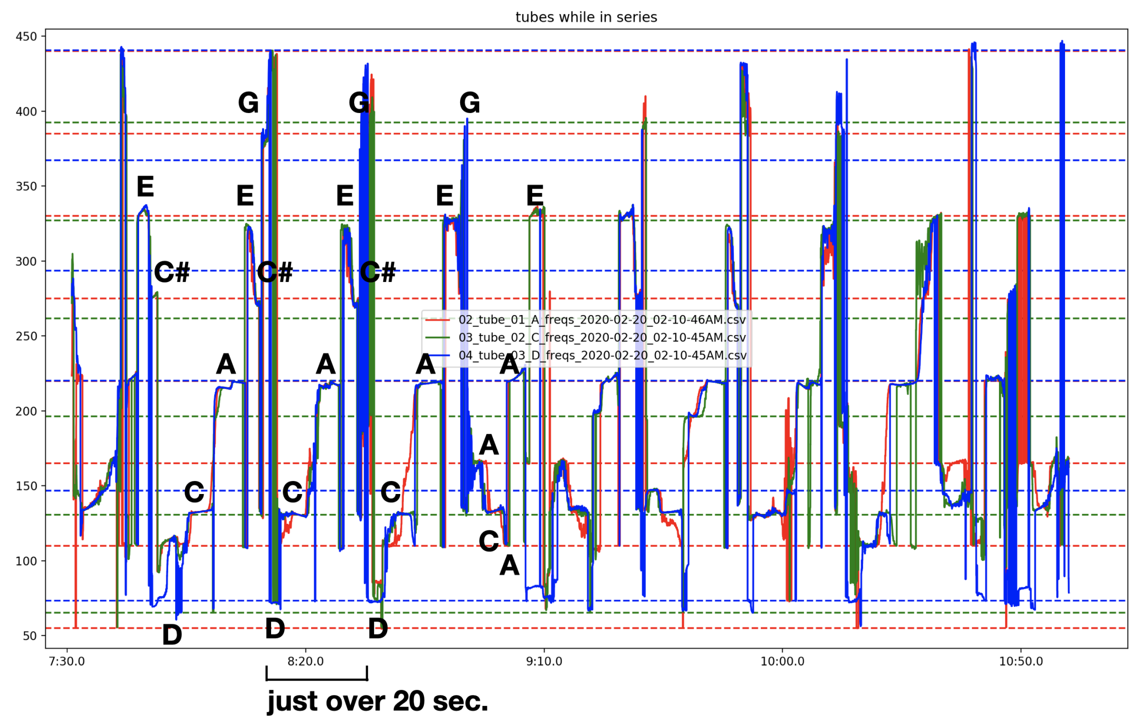


Figure 13: All three tubes frequencies though time, showing periodic repetition of certain frequencies.

Tube Interaction Zooming in on the tubes' frequency plots displays more complex interactions. Figure 14 shows how a dip in the sounding frequency of the system is passed around the feedback loop through each tube. At 8:08, all three tubes are around 325 Hz, then transition to about 275 Hz by 8:09.6. During this descent, the tubes' frequencies deviate from each other slightly, revealing a dip in frequency that cycles through tubes: A, then C, then D.

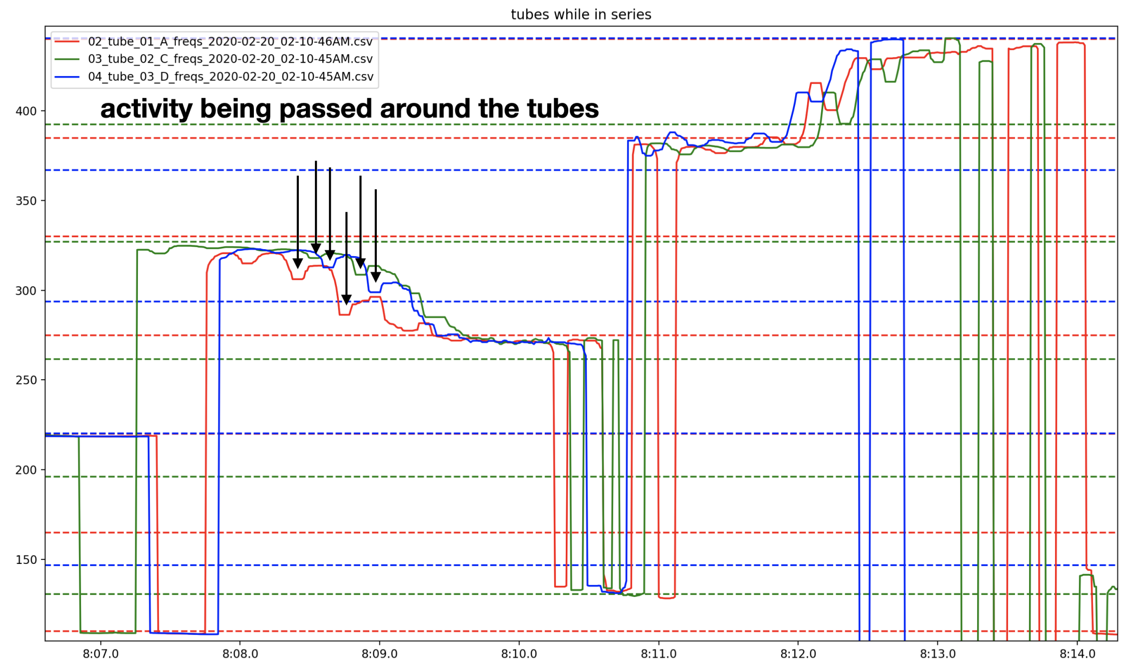


Figure 14: Frequency changes in the three-tube feedback system being passed around the tubes.

The drastic and jittery deviations between partials seen in these graphs can be understood by Figure 15, which plots a sonogram of the tube's audio recording and overlayed with the pitch tracker line analysis. This shows that the monophonic pitch tracker is responding to other frequencies present in the tube, yet is mostly representative of the tubes' (and system's) strongest resonances. Aural perception of the system reinforces the presence of this polyphony.

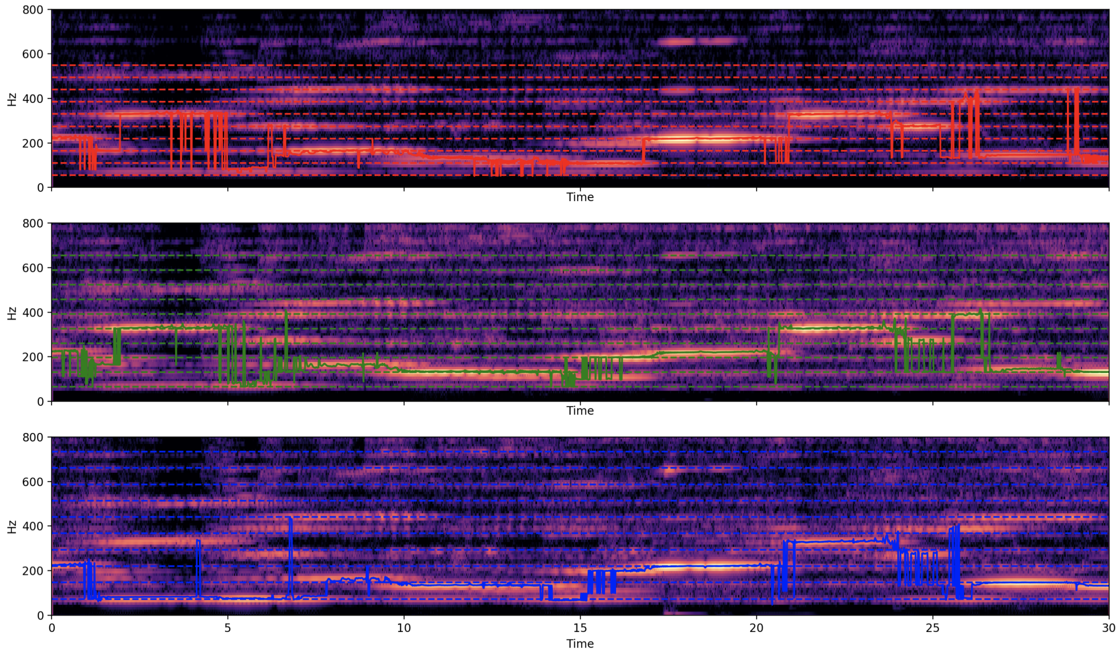


Figure 15: Sonogram of each tube with pitch tracker overlayed.

Figure 16 shows more complex interactions, including (1) how individual tubes can lead the system to one of its own partials, “dragging” the other tubes to join, (2) interesting divergences where one tube will resonate at a frequency very different from the others, and (3) transitional spaces where no clear stability can be observed. Figure 17 shows (1) a moment where the A tube (red) stays steadily on its own partial while the C (green) and D (blue) tubes seem to be “fighting” with each other for which harmonic series to come to rest in and (2) a moment when the system transitions from the second partial of C (green dotted line) to the third partial of A (red dotted line), however, the D tube (blue) seems to resist this motion, attempting to remain at its second partial (blue dotted line) as the system passes by that frequency.

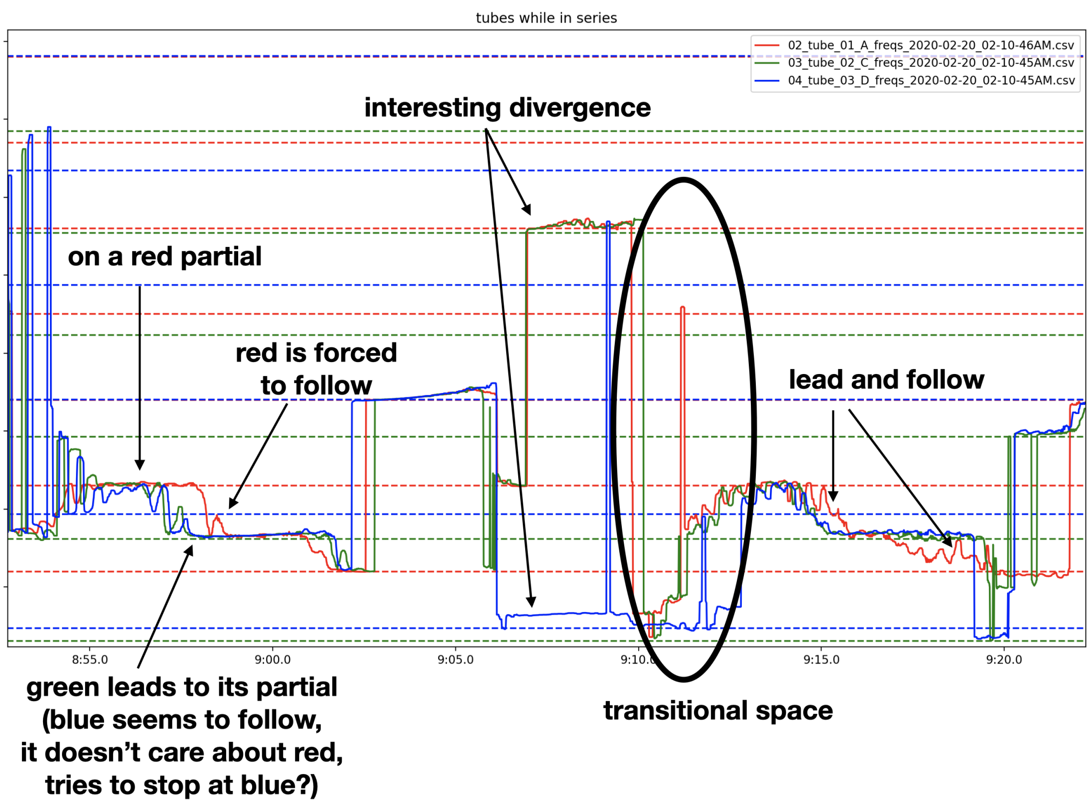


Figure 16: More tube interactions.

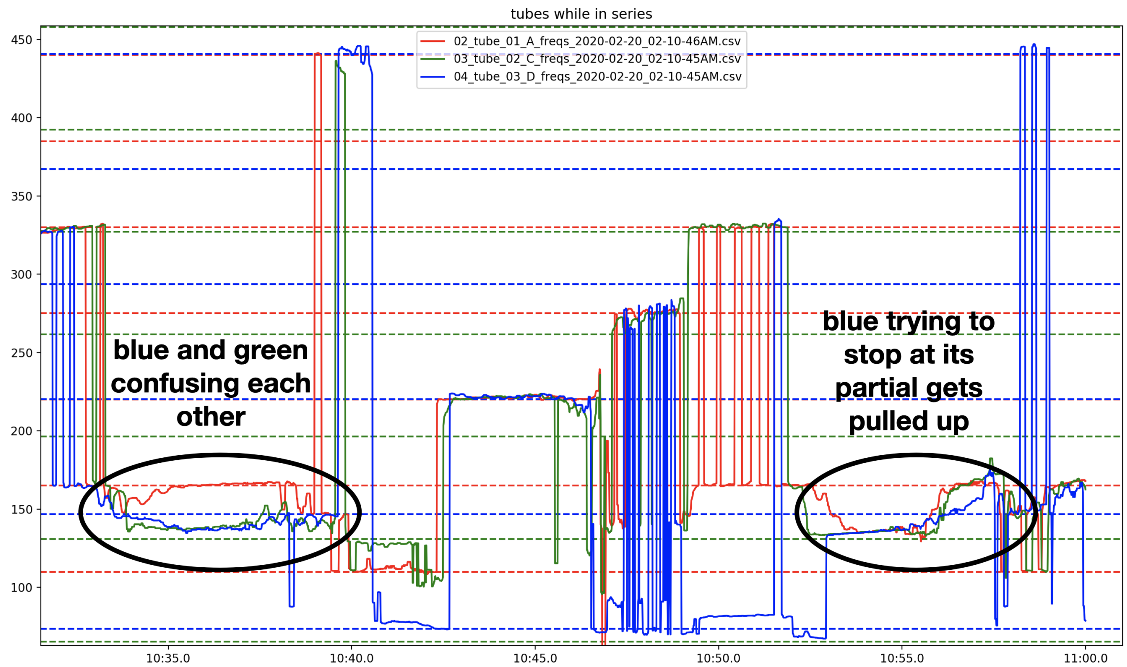


Figure 17: More tube interactions.

Uncovering Battle Grounds The transitional moments and ensuing “fights” between tubes are the most sonically compelling passages that arise in performance. By further understanding these moments in particular, I hope explore their possibilities and perhaps identify strategies to induce them in other feedback systems. Figures 18, 19, and 20 show two-dimensional histograms of the relation between two tubes’ frequencies while in series. Each point is a moment in time indicating the simultaneous frequency of the two tubes being represented. Bluer points represent more moments in time; a “taller” peak on the histogram. These plots show where in frequency space the two tubes tend to be–more dense clusters represent more time spent in that state. The diagonal line at $y = x$ (or about 45°) represents unisons, where both tubes are at the same frequency. Diagonal lines fanning out from the unison line are integer multiple relations, which would indicate that the tubes are not at the same frequency, but in harmonic relation with each other.

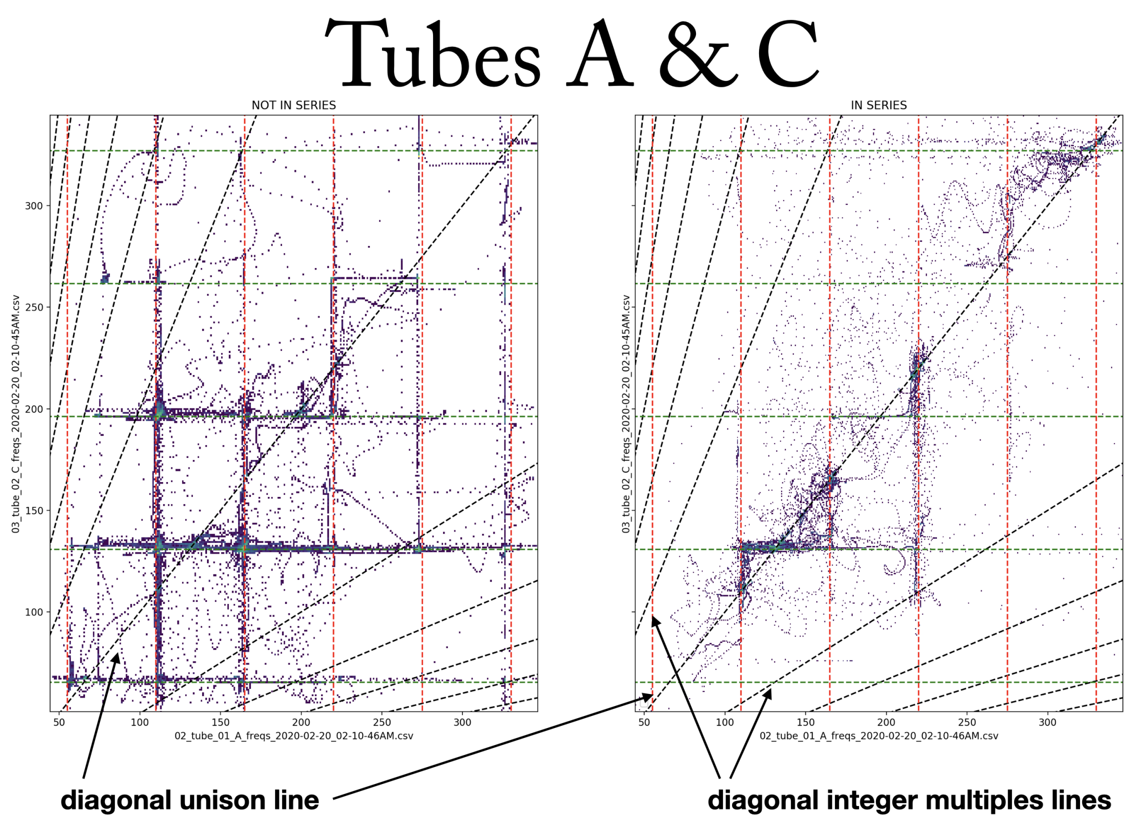


Figure 18: 2D histogram of frequencies in A and C tubes.

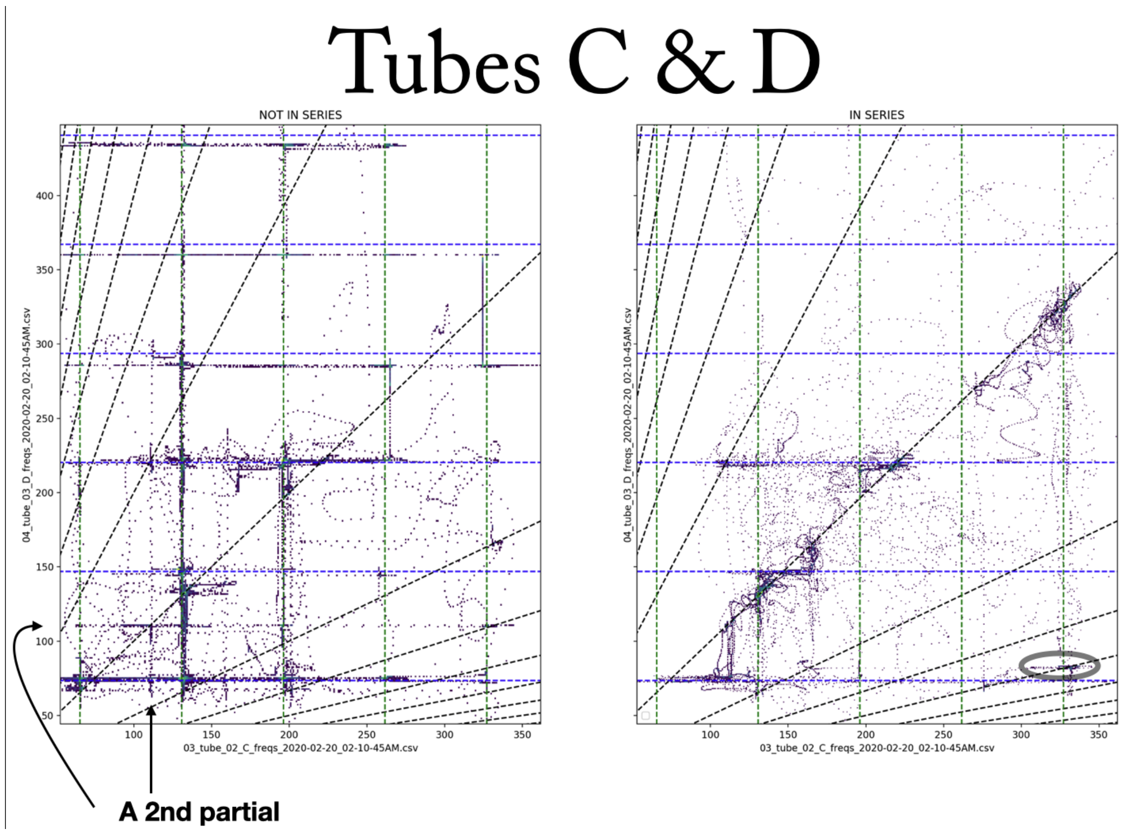


Figure 19: 2D histogram of frequencies in C and D tubes. Gray circle represents the stable state seen in Figures 24 and 25

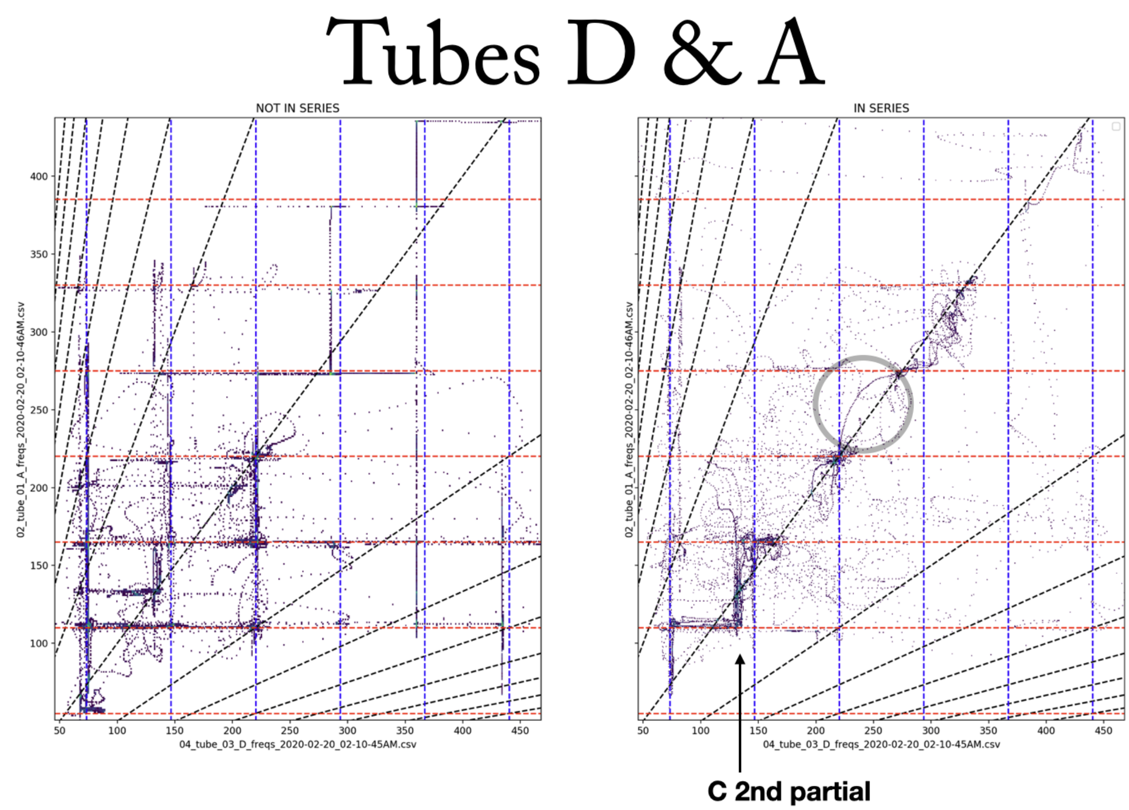


Figure 20: 2D histogram of frequencies in D and A tubes.

From these plots, one can see that while the tubes are not in series (left side) each tube is independent, mostly sounding its own partials, as expected. While the tubes are in series (right side), however, the points, or "states," are much more clustered around the unison line, as the system is one feedback loop, sounding (mostly) one resonance. Also, there are clear clusters of states at points along the unison line, indicating locations of stability (stable states, or homeostasis) that the system prefers to resonate at. One can also see looping curves through the space, which can be assumed to be connected into lines through time representing a transition from one stable state to another in which one tube resists leaving its own partial, but eventually is "dragged along" to a stable state not in its harmonic series. For example, the circled arc in Figure 20 shows the system transition from the A tube's fourth partial (which is also D tube's third partial), to A tube's fifth partial, however, the D tube clearly resists this motion initially (the point is trying to maintain its x axis position, therefore the arc starts by moving up instead of diagonal along the unison line). The resonance in the D tube eventually succumbs to the system's movement, allowing the x position of the point to move to the right, reconnecting with the unison line (the tubes are again in unison) where it intersects the A tube's fifth partial.

The harmonic content of the listening experience is reflected in the position of stable states, which outline an A Mixolydian scale with an added C natural in the lower register. There is also a strong subdominant presence in the tubes' performance, created by the D tube. Comparing Figures 9 (histogram of each tube's sounding frequencies while in parallel) and 21 (histogram of all tubes combined while in series), one sees that the scale of the three-tube feedback system is a combination of some partials from all the tubes. It is important to notice however that although the unison line crosses all possible partials, there are not point clusters at all crossings; there are some partials that the three-tube feedback system does not come to rest at (i.e., resonate at).

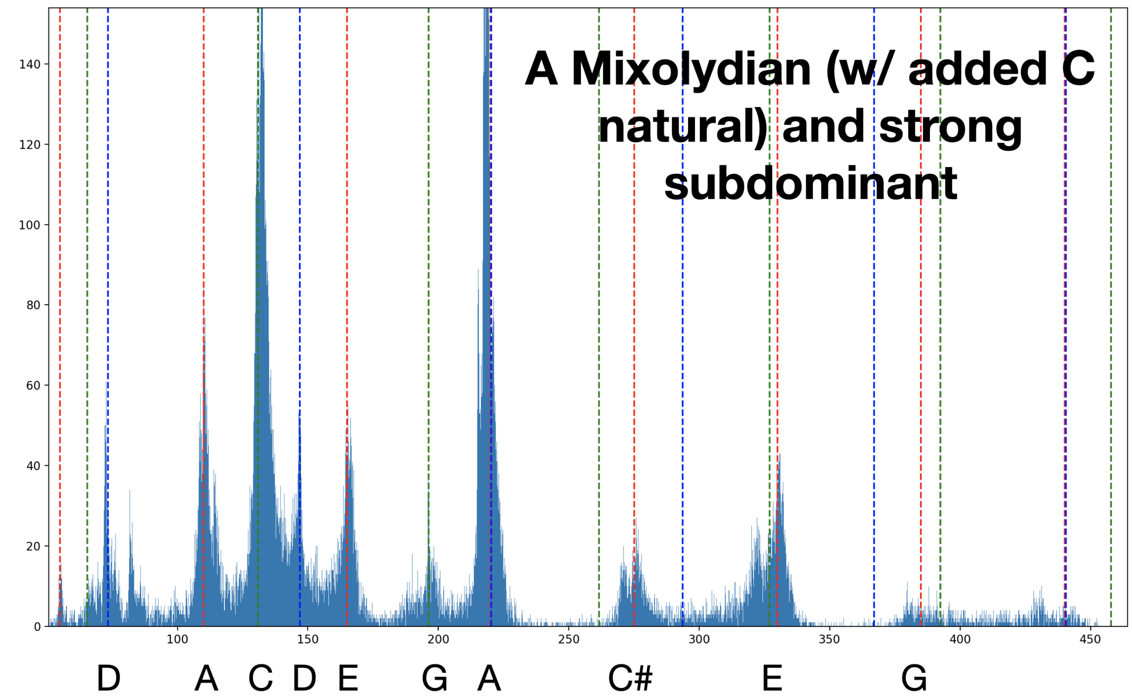


Figure 21: Histogram of all three tubes' (in series) frequency analyses combined.

In order to analyze how the resonance tendencies of the whole system relate to the different harmonic series of the tubes, Figure 22 show the normalized distance of each tube's sounding frequency from its nearest partial (a value of 0.0 distance means it is at a partial in its harmonic series, 0.5 means it is directly in between two of its partials). For any given point in time, this plot clearly shows which tubes are resonating within their harmonic series (close to 0.0) and which tubes are not (close to 0.5). Figure 22 shows a moment that transitions from a stable state only near a C tube partial to a stable state only near partials in the D and A harmonic series. This is easy to see in the bottom graph (the pitch of each tube) as well as the top graph (each tube's relative proximity to it's nearest partial). Most stable states are at frequencies that allow two of the tubes to resonate in their harmonic series (there are none that encompass all three); two examples can be seen in Figure 23. While all these examples lie along the unison line, there are some states in the system that seem to be stable, yet are distant from the unison line, for example the same stable state is seen in Figures 24 and 25. The circled area in Figure 19 shows on the two-dimensional histogram the cluster of points very distant from the unison line representing this stable state.

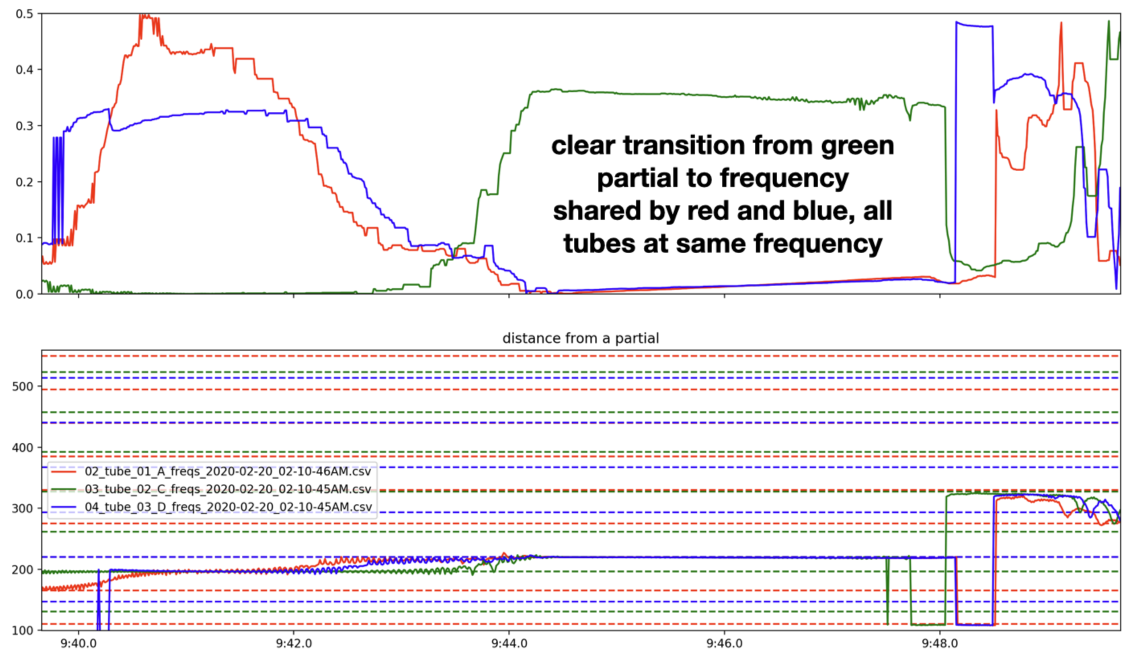


Figure 22: A transitional moment demonstrated by the distance of each tube's sound frequency from its nearest partial.

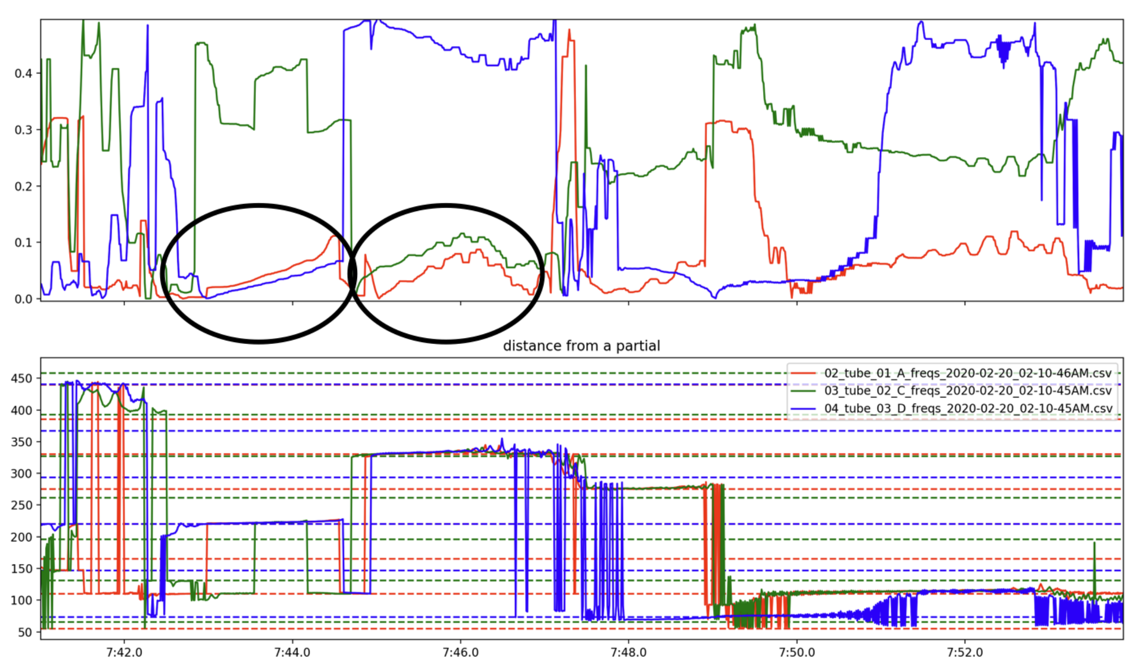


Figure 23: Two stable states, each of which is near a partial in two of the harmonic series.

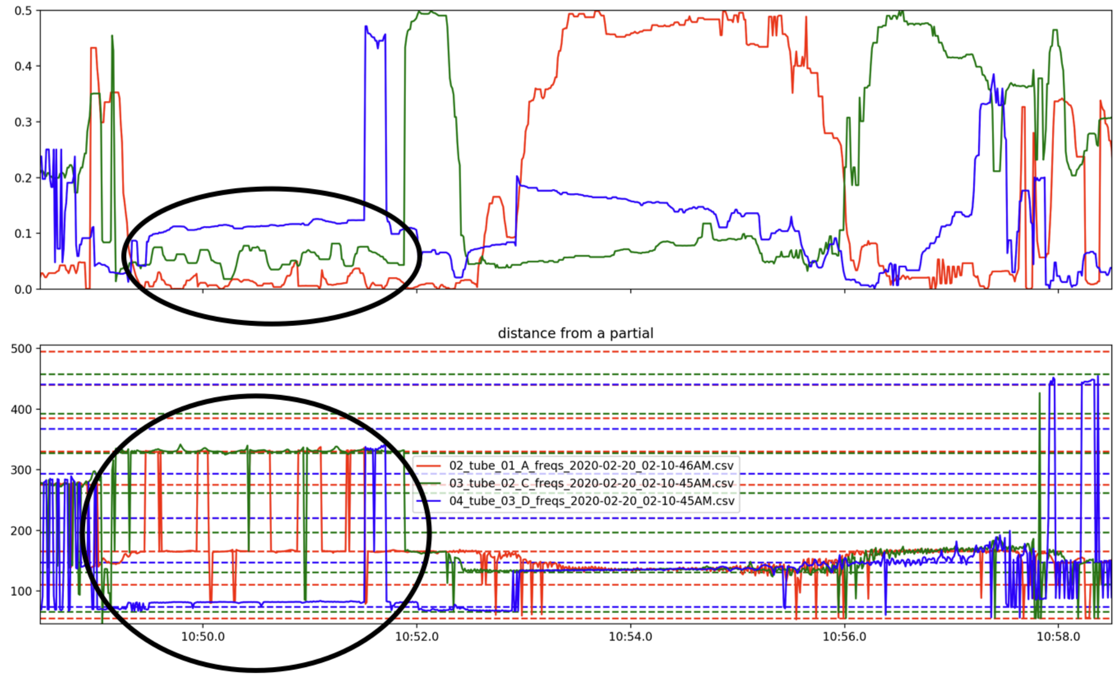


Figure 24: A stable state distant from the unison line.

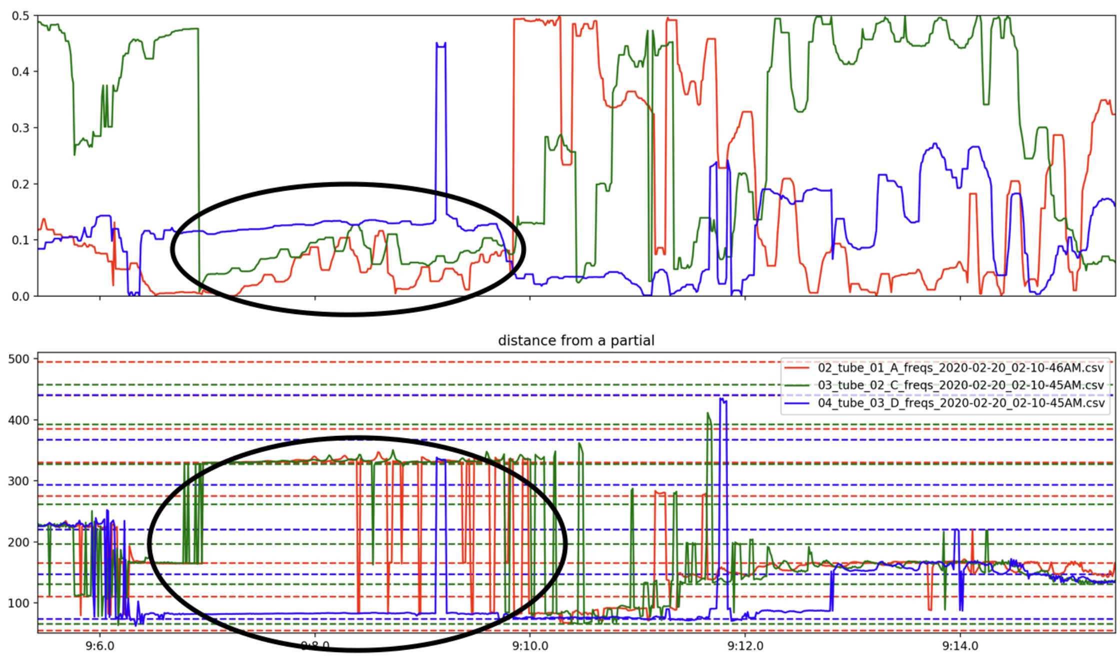


Figure 25: A stable state distant from the unison line.

The final plotting strategy used to understand this system shows clear "battle grounds" where the tubes are "fighting" over which of the system's stable states (most of which are in A Mixolydian) to settle on. Figure 26 shows the relation of C tube's (green dots) and D tube's (blue dots) frequencies (y axis) in relation to A tube's frequencies (x axis) (red dots are the relation between A tube and A tube, therefore always on the unison line). The large black dots on the unison line represent the stable states of the system (as seen in Figure 21). One can again see that some partials of tubes are not included as stable states (such as the fourth and fifth partials of D and fourth partial of C). More interestingly, one can see square-shaped clusters of points that use adjacent stable states as the bottom-left and top-right vertices (those on the unison line). Figure 27 shows larger square shapes created by non-adjacent stable states. These squares show a lot of structured activity near these stable states as the system transitions between them. The curved white line seen in Figure 27 (which is the same one in Figure 20 seen from a different angle), again shows the D tube attempting to remain at its third partial on the y axis (pitch A), while the system moves to the C# above it, eventually curving up and also arriving at C#. I refer to these squares as "battle grounds" because they represent the pitch space in which the system is out of homeostasis as the three tubes seemingly "battle" for the system to settle at a state that is within their harmonic series.

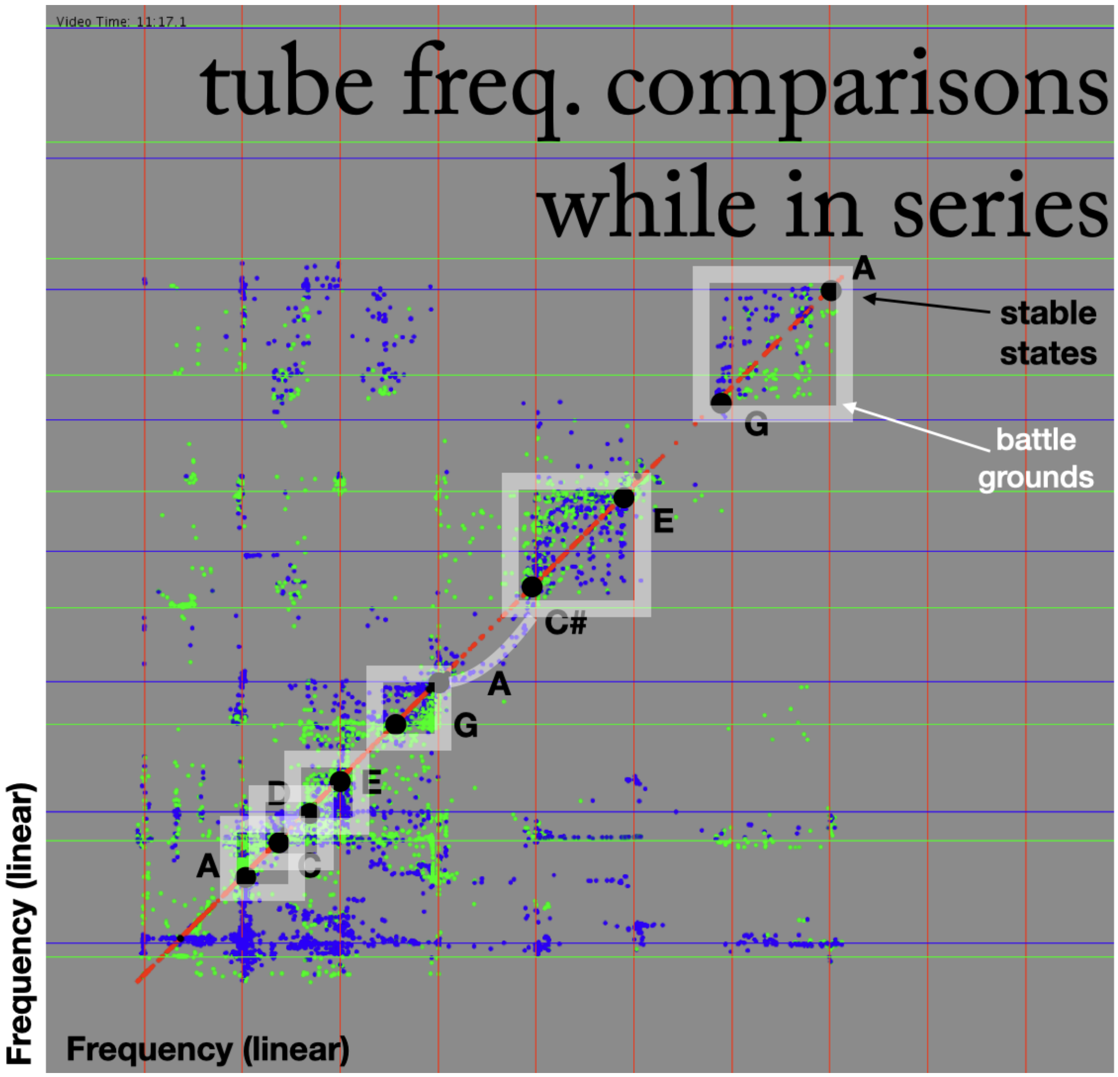


Figure 26: Frequency of C and D tubes in relation to A tube and appearance of “battle grounds.”

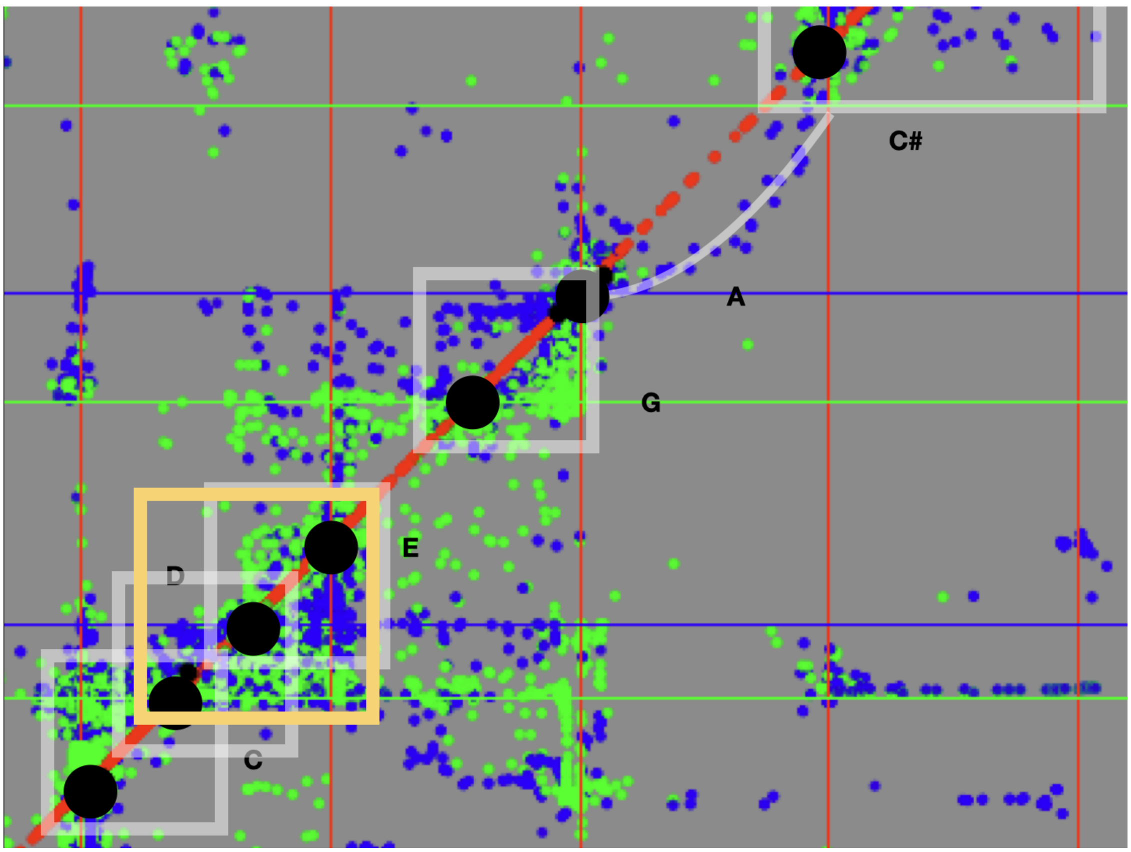


Figure 27: Zoomed in region of "battle grounds" revealing a larger battle ground created by the non-adjacent stable states C natural and E natural

The “battle grounds” shown in Figures 26 and 27 can also be seen (without white boxes) in Figures 28 and 29, as well as other seemingly non-arbitrary structural shapes further off the unison line. Video representations of these plots, which more clearly demonstrate the “battles” as they occur through time, can be viewed here.[15]

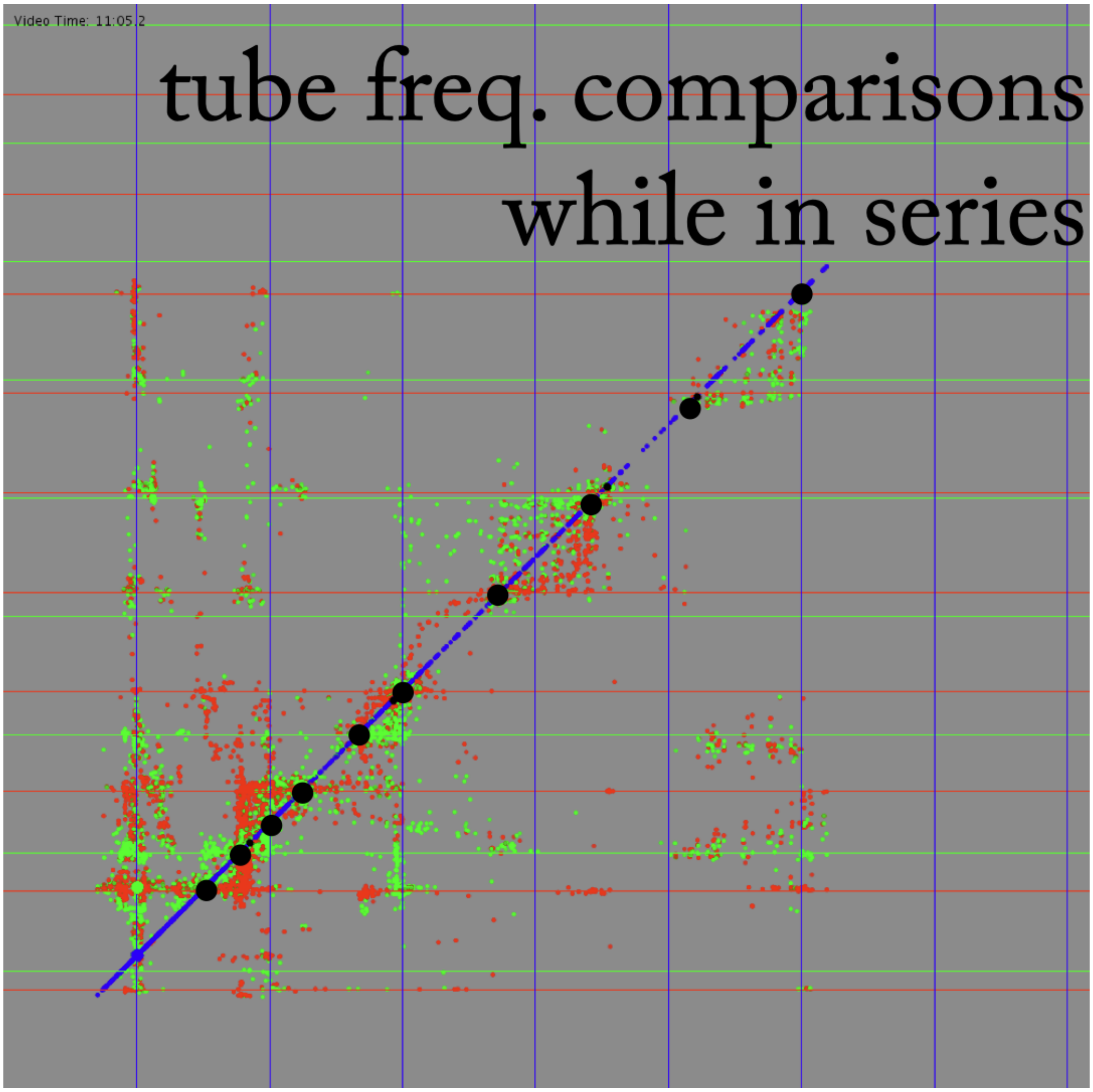


Figure 28: “Battle grounds” seen by comparing A and C tubes.

[15] https://drive.google.com/drive/folders/1vAHhJI5Jp32hrM_FucdQKCzSMwHouYGU?usp=sharing

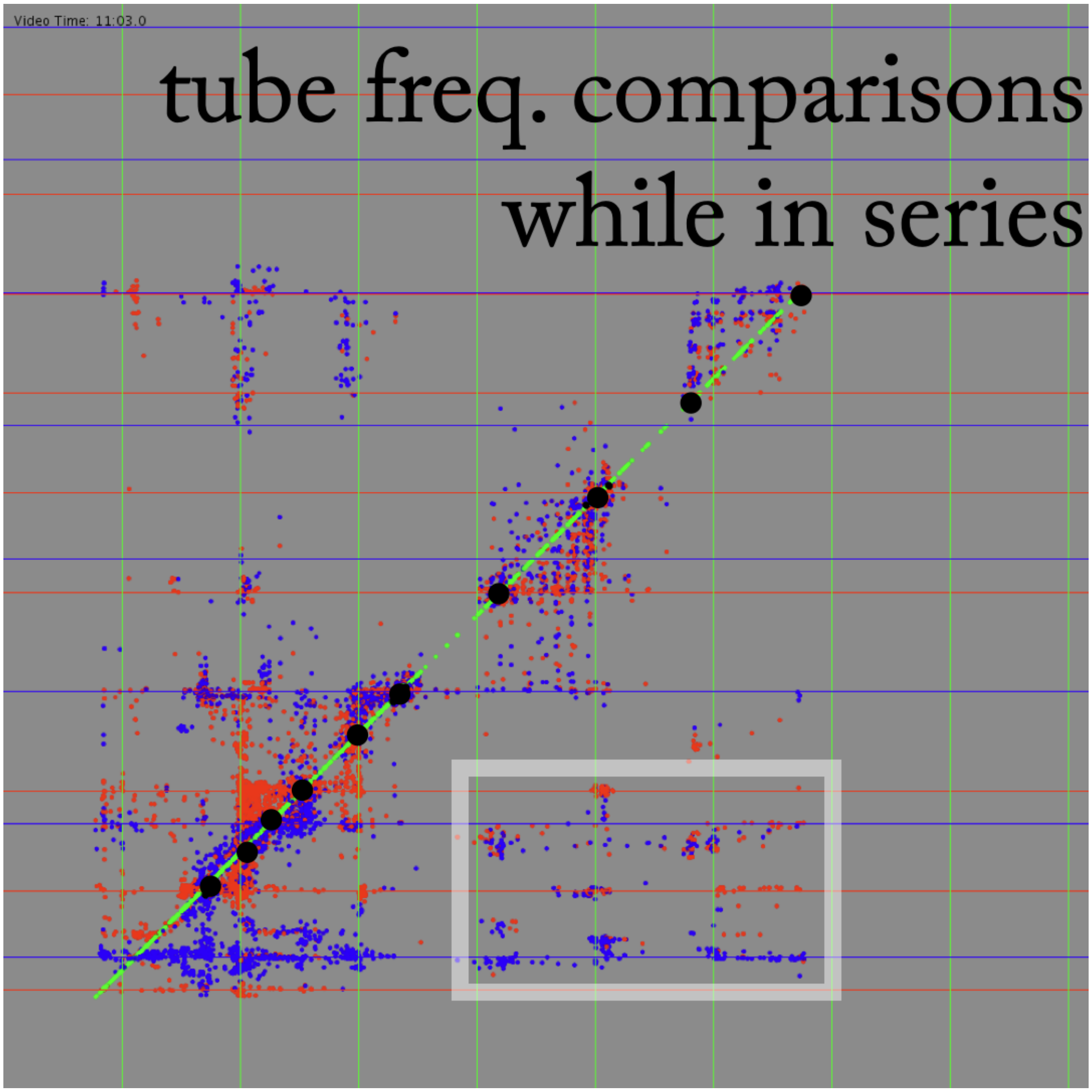


Figure 29: "Battle grounds" seen by comparing A and D tubes.

Feedback Saxophone The saxophonist is also performing a feedback loop created by placing a small lapel microphone in the neck of the instrument (the mouthpiece is removed). The sound transduced by this microphone is amplified through the house speakers (not speakers used for the tubes) creating a feedback loop that is responsive to the resonance of the saxophone body. Before performing with the whole saxophone body, the microphone is first placed in only the neck, creating a much smaller tube for resonance. Once the entire saxophone body is attached, the resonanting length of tube can be manipulated by the keys, changing the pitch and offering performativity to the saxophonist. Figures 30 and 31 are histograms of the resonance of each of these systems (as analyzed from the performance audio) which shows more diversity of frequency content than any single tube, but also is clearly influenced by the tubes' resonances–often sounding frequencies that are in in the tubes' harmonic series.

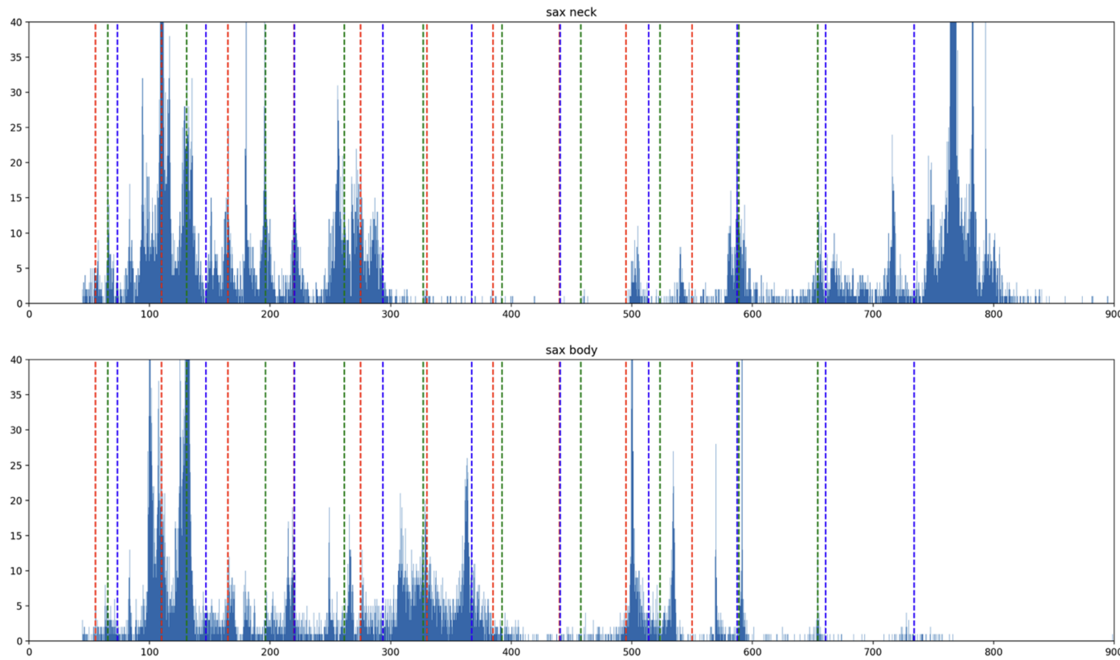


Figure 30: Resonances histograms of saxopone neck (top) and full construction (bottom) as heard in *hollow*.

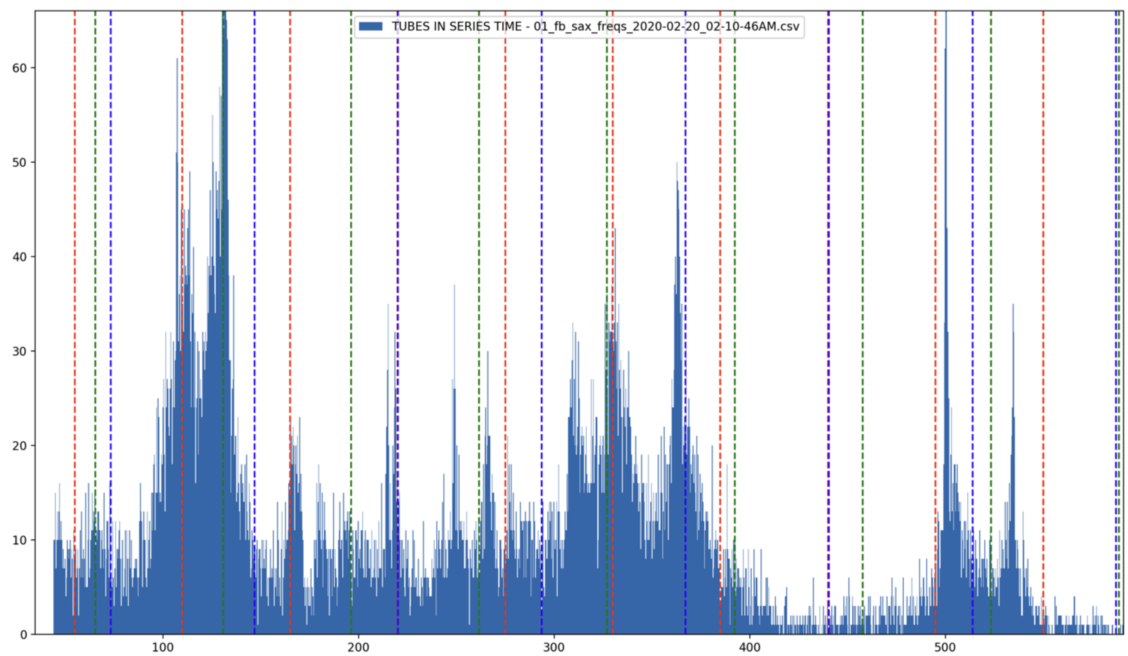


Figure 31: Resonance histogram of the saxopone (full construction) while the tubes are in series.

Analyzing the saxophone's sounding frequency through time (seen in Figure 32) again reveals that it is often resonating at frequencies found in the tubes' overtone series. Taking a shorter excerpt one can identify specific frequencies and their relation to specific tubes (Figure 33, this excerpt can be heard here[16]). Comparing the frequency motion of the saxophone at this moment to that of the tubes (Figure 34) reveals that pitch trackers of the saxophone and C tube are reporting the same frequency curve as both transition from around 270 Hz to 220 Hz. Although one cannot be sure if one was simply "hearing" the other, or if the two signals were truly influencing each other, this does show that the two feedback systems (i.e., tubes feedback loop and saxophone feedback loop) were sonically aware of and potentially actively influencing each other. This hypothesis is strengthened by Figures 35, 36, and 37 which show simultaneous frequencies of the saxophone and each tube respectively. One can see that not only does the saxophone spend a lot of time on the unison line with each tube, but also that when not on the unison line, the saxophone is often on an integer multiple line, indicating that the saxophone is in some harmonic relationship to the tube's sounding frequency.

[16]`https://vimeo.com/382472269` 6:34-6:42

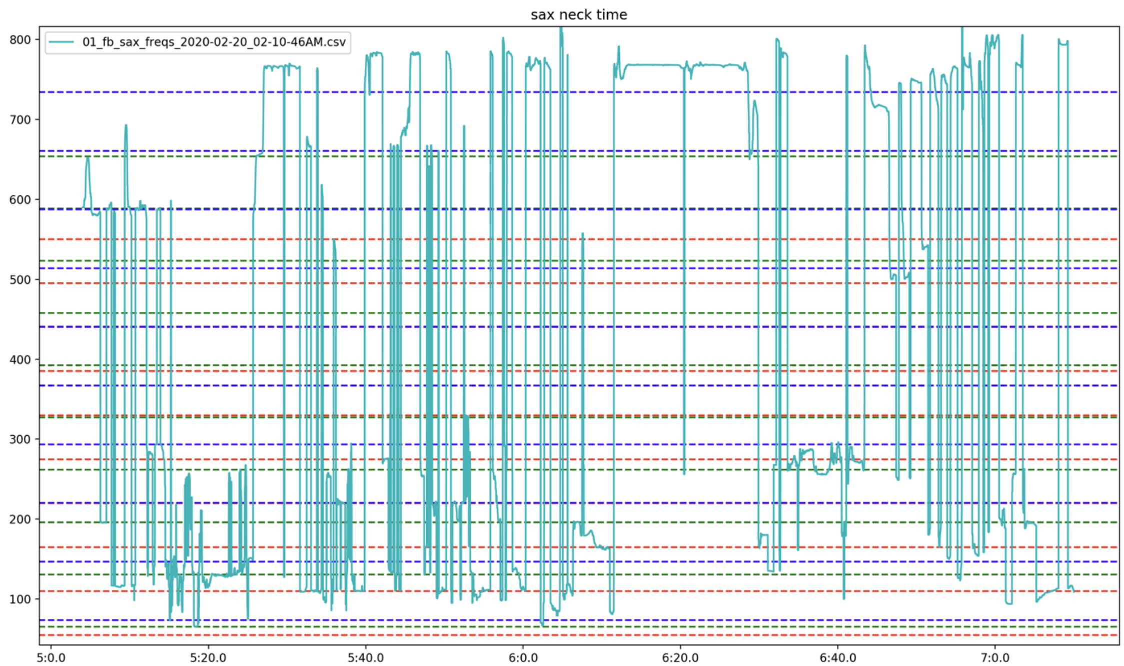


Figure 32: Frequency of saxophone audio analysis in relation to tubes' harmonic series, while being performed with only the neck.

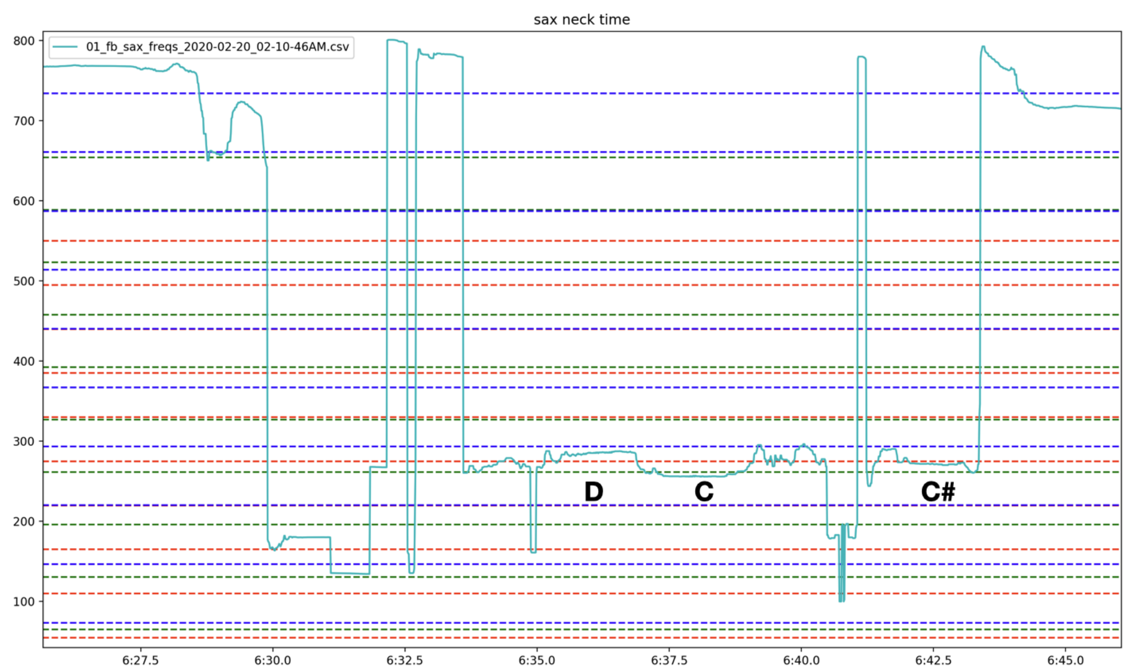


Figure 33: Excerpt of saxophone frequencies showing clear pitches in relation to tubes harmonic series.

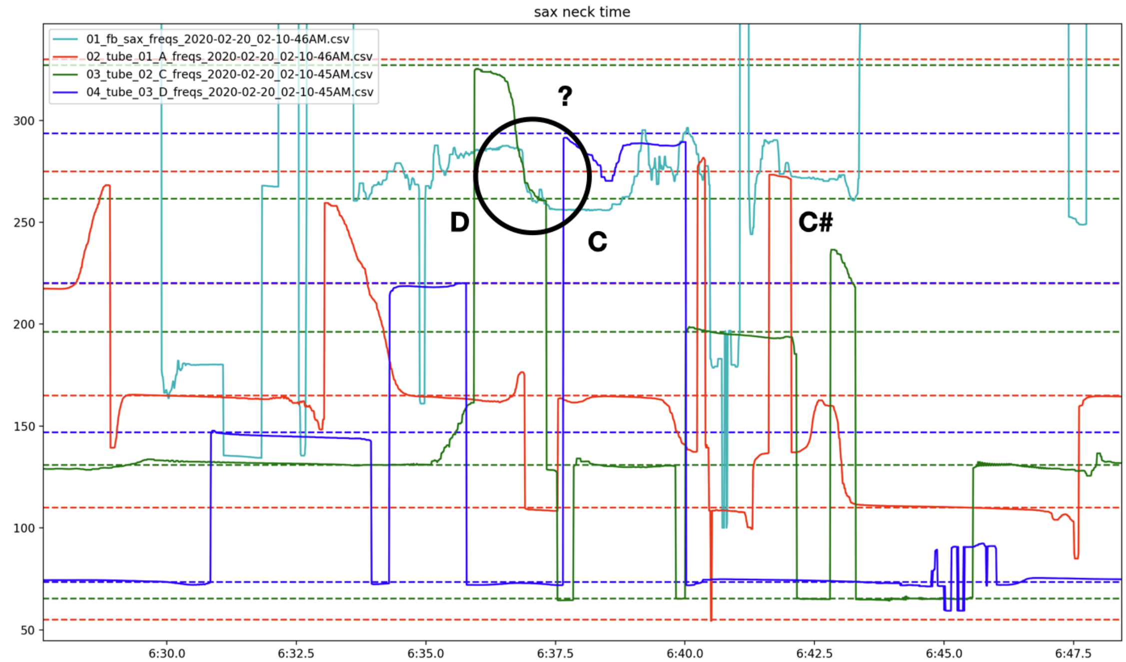


Figure 34: Unison motion between saxophone and C tube analysis frequencies.

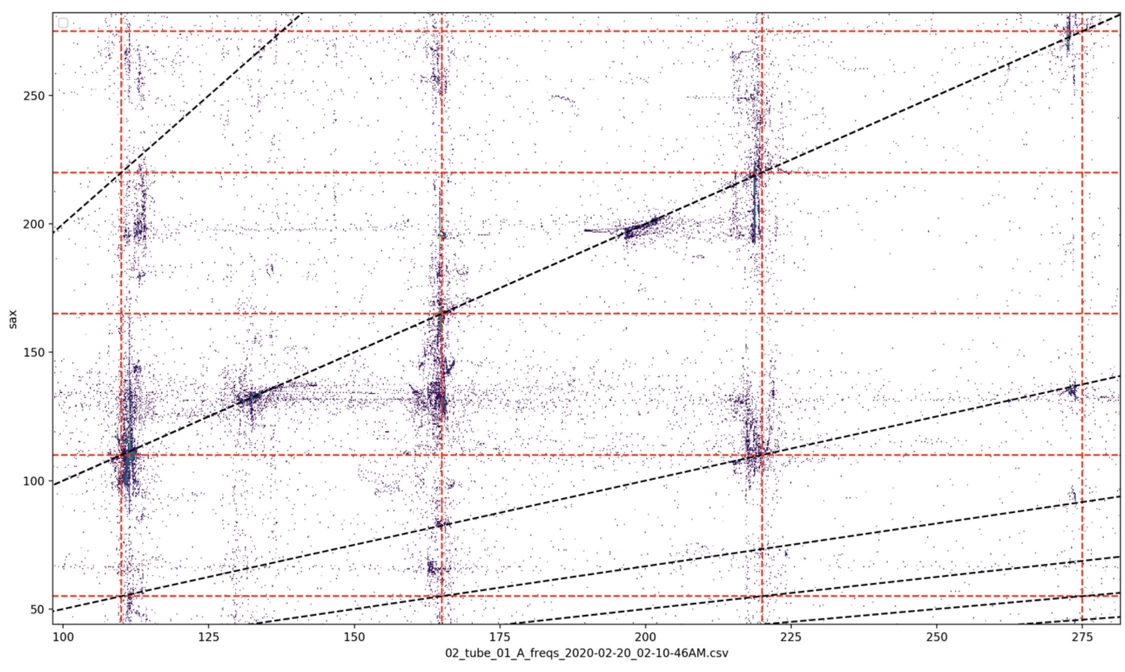


Figure 35: Two-dimensional histogram of sax and A tube analysis frequencies.

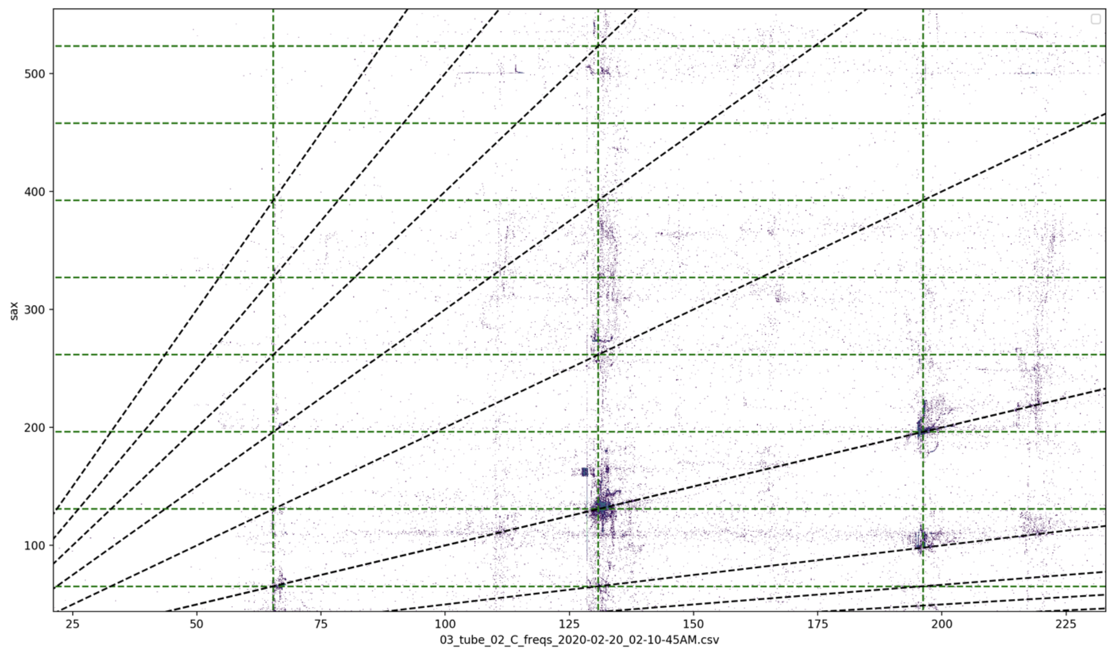


Figure 36: Two-dimensional histogram of sax and C tube analysis frequencies.

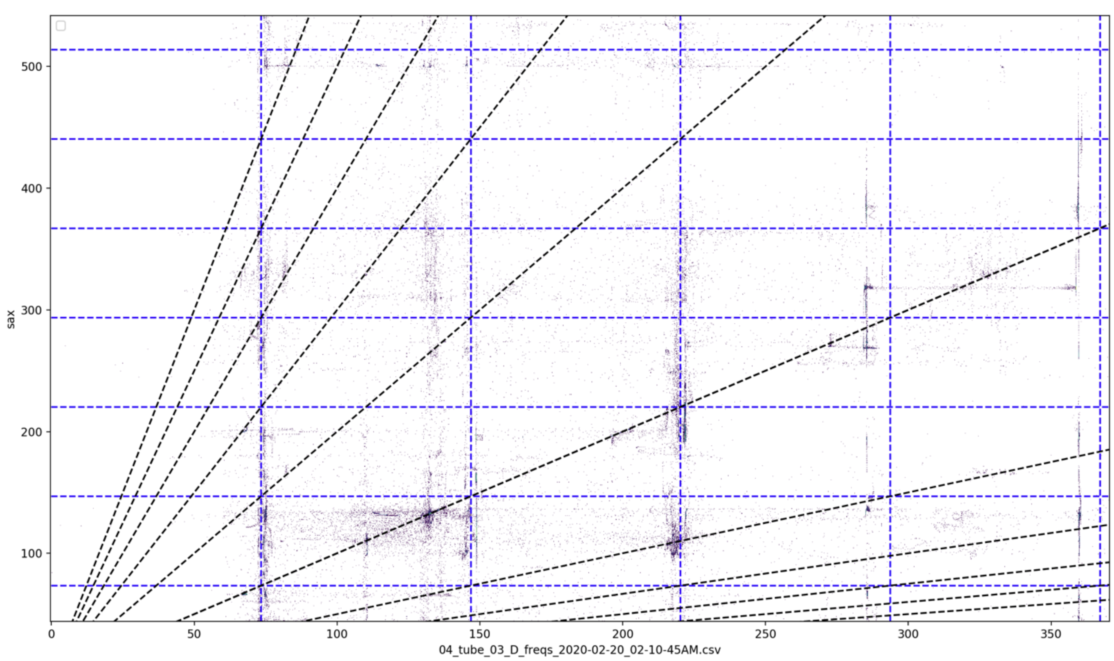


Figure 37: Two-dimensional histogram of sax and D tube analysis frequencies.

Conclusion When connected in series as one large feedback loop, the tubes act as filters, which interact with the modulated delay lines and resonance suppressors to create a performance based on a scale similar to A Mixolydian (created through a combination of frequencies based on the tubes' lengths). The scale's pitches represent the system's most stable states as heard and seen through analysis. "Battle grounds" between stable states are areas of activity created by different tubes trying to persuade the system to settle on a stable unison frequency that is included in their harmonic series. Stochastic elements in the system probably cause (or obscure the cause of) certain behaviors, such as the cycling of certain frequency patterns (Figure 13). Some questions remain unanswered, such as why some tube partials are not included as a stable state and included in the emergent scale.

The analytic approach taken and tools developed are able to reveal the structure found in the stability and instability of the system. Hopefully these tools can be applied to other feedback systems for similar results.

3.3.2 Creating Hamiltonian Paths for Phrase and Form Generation

Using "organizing sounds in time" as a definition for compostion frames the process around two questions: (1) what are the sounds that are being organized? and (2) what is the strategy for organizing them? Algorithmic composition has a long history of answering the latter and has mostly been focused on the manipulation of common practice musical objects, such as pitch, note, rhythm, meter, harmony, etc. (Nierhaus 2009) Recently, AI and machine learning have also been used to analyze and compose music using these materials as datasets. (Kotecha and Young 2018) (Eck and Schmidhuber 2002) As an electronic musician I am interested in organizing (and algorithmically manipulating) sound objects from a more Schaefferian perspective. (Schaeffer 2017) In this study the sounds I set out to organize are a database of 100 millisecond audio samples and the strategy used to organize them is a Hamiltonian Paths created with (1) an algorithm for solving the Traveling Salesperson Problem and (2) UMAP dimensionality reduction (McInnes, Healy, and Melville 2018).

In mathematical graph theory, a Hamiltonian Path is a path through a graph that visits each node only once. A classic computer science problem that uses a Hamiltonian Path is the Traveling Salesperson Problem which aims to minimize the distance of the Hamiltonian Path, thereby minimzing the distance a hypothetical salesperson needs to travel on their tour visiting many cities (graph nodes). Using my database of audio samples as the nodes of the graph, a Hamiltonian Path that connects them all into a linear organization, which for musical purposes can then be used as a temporal organization.

My initial investigations in this strategy used samples of saxohpone, bassoon, drums, bells, no-input mixer, and synthesizer sounds.[17] Each audio file was split into 50 millisecond slices and analyzed on 23 parameters in SuperCollider.[18] This dataset was then imported into Python, normalized, and processed with the Python library tsp-solver (Goulart 2021), which attempts to find an optimal (or near optimal) sequence of datapoints that create the shortest path.[19] This sequence is then reimported to SuperCollider and synthesized in non-real time, by copying each 50 millisecond slice into the sequenced order using an overlap of two and a triangle window. The source audio files and results can be heard here.[20]

Timbral Fusion Sequencing the audio samples by minimizing the Hamiltonian Path's distance creates a temporal organization in which one can hear the algorithm "tour" all the different audio sample nodes. There are clear moments of touring through the "saxophone", followed by the "no-input mixer space", etc. However, because some saxophone and no input mixer sounds will be similar, the path will often interweave similar sounds from different sources in indentifiable ways. One can hear many examples of this the resulting soundfile,[21] such as at 3:15-3:18 when an ascending saxophone line is followed by an ascending line from the no-input mixer and seamlessly handed off back to the saxophone. The algorithm has identified that these timbres are similar and therefore has placed them in sequence on the Hamiltonian path.

Phrase / Gesture Creation There are also many moments throughout the recording that I hear as musical phrases or gestures, the first is from 0:29-0:34. After a 29 second quiet, sustained introduction, the composition (i.e., the Hamiltonian Path) suddently has a five second outburst that starts with a pure high tone, followed by some noisy moments and a short three note motive, before returning to quiet sustained timbres. A moment later from 0:38-0:39 a crescendo occurs (officially ending the sustained introduction, getting into the B section of the work) that has a beautiful gestural shape. Over the course of the one second, the loudness increases, the timbre becomes less noisy, the morphology becomes less static, the panning becomes less mono, and the bass register enters at the end of the crescendo. This parametric gesture and the preceding phrase make for compelling musical expressions, the kind that I might have created through improvisation or composing sounds in a DAW, but has emerged from this Hamiltonian

[17] These files can be heard here: `https://drive.google.com/drive/folders/1MYfiRtb4MdCsUKiEpiWjcFl0quV8Fm4w?usp=sharing`

[18] amplitude, spectral crest, spectral slope, spectral spread, loudness, sensory dissonance, spectral centroid, spectral flatness, spectral percentile, zero crossing, MFCC 1-13

[19] Since it is not possible to know if the optmial path has been found, the solver stops at some epsilon criteria where it seems to be no longer improving its solution.

[20] `https://drive.google.com/drive/folders/1MYfiRtb4MdCsUKiEpiWjcFl0quV8Fm4w?usp=sharing`

[21] `https://drive.google.com/file/d/1olSVpNBmBlZUynGLfKIKXEy6klcLxXWu/view?usp=sharing`

Path. In a moment like this I, the composer, experience mirroring with the algorithm. The agency of the algorithm becomes visible and my collaborator is established.

It's interesting to note that while one can often identify the similarities in sound (and therefore temporal organization) the algorithm is "trying" to create one can also hear where a juxtaposition seems, not like a timbral fusion, but out of place; where the algorithm seems to have made a strange choice or error. One such moment is the sudden saxophone timbre at 0:49, otherwise surrounded by 13 seconds of bells. While a version with fewer of these "out of place" moments might sound like a more "successful" (i.e., shorter) path, I hear these "deviations" as compositionally pleasing ornamentation, or recurring motives that tie the work together. These moments help me perceive agency in the algorithm because they are surprising. Their conspicuousness seems to convey intention. An output without any surprises such as these might sound like the cold, calculated output of an algorithm, not one that conveys intent and therefore agency.

Form While I don't think that the resulting soundfile works formally as a finsihed composition, there are some intersting emergent formal properties. The most obvious one is the long sustained introduction. The Traveling Salesperson solver algorithm starts its path at a random point, so it is by chance that this section became the introduction, however even if it was elsewhere it would still likely have clustered these sounds together in time because they are so different from anything else in the dataset (they are a distant island of points in the 23 dimensional space). Other emergent forms include the alternating of phrases, moving between different sound sources, often every 3 to 10 seconds. These phrase alternations also appear with transitional cues where it seems the path has had to traverse across one space to get to the next, such as the synthesiser sounds from 1:56-1:57 that act as a transition from saxophone sounds to drum sounds. Figure 38 shows the relationship between different positions in the input files and their position in the output file. All the source soundfiles are represented on the y axis, not in any particular order. The start of each file is at the bottom of it's horizontal bar, the end at the top. The x axis represents the duration of the output file, the beginning on the left, the end on the right. The red line represents $y = x$, so any shapes that appear parallel to it have occured in the output file in the same temporal ordering as in their source, as can be seen and heard during the introduction, which uses mostly sounds from the "sustain_elec 01" sound file. The musical material that was generated with this algorithm was ultimately used in a tape part for my work *squall*,[22] by slicing out phrases and moments that I found most successful.

[22] https://drive.google.com/file/d/1vlMoSnKt3oP6ZqjnVB4tACec-_PUzCwW/view?usp=sharing

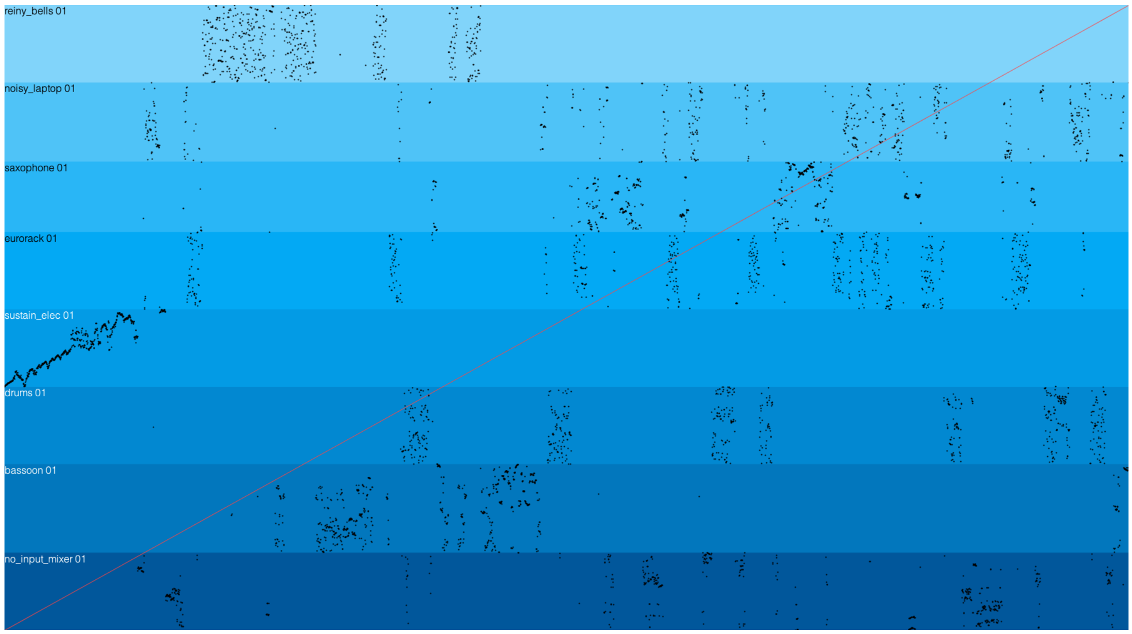


Figure 38: Comparative Matrix of sound source positions in original file (y axis) and positions in minimal distance Hamiltonian Path (x axis) for sound sources: saxohpone, bassoon, drums, bells, no-input mixer, and synthesizer.

Comparing TSP and UMAP Another dataset on which I used this strategy consists of sound samples of flute, cello, piano, and mixed percussion. The piece was commissioned by the Switch Ensemble for the 2021 SEAMUS Online Conference. After composing around two minutes of music for the performers to audio and video record, I analyzed the audio tracks in 100 millisecond[23] slices using 51 analysis parameters.[24] From the nine FFT frames in each 100 millisecond slice I extracted statistics on the analysis parameters and their first derivates, which in total created an analysis vector of 714 dimensions. Using principle component analysis on this dataset enabled me to reduce the number of dimensions to 11 while retaining 99% of the variance.

Using the 11 principle components as my new analysis vectors I first used the same tsp-solver (Goulart 2021) algorithm as above to create a minimal distance Hamiltonian path of these sounds. I then recombined the sequence of 100 millisecond sound slices (and their video frames) into a new file.[25] Figure 39 again shows where each slice in the final path came from in its respective source file.[26] Because the act

[23]For this implementation I chose 100 millisecond slices rather than 50 milliseconds because there was video involved. At 50 milliseconds, there would most often only be one video frame associated with each slice. With 100 millisecond slices, there will most often be 3 frames per slice, making the video aspect of the experience more relevant.

[24]spectral centroid, spectral spread, spectral skewness, spectral kurtosis, spectral rolloff, spectral flatness, spectral crest, pitch, pitch confidence, loudness, true peak, and 40 MFCCs; for FFT analysis the window size was 1024 with a hop size of 512

[25]`https://drive.google.com/file/d/1Iz3J5ayEJa2OMM3YIPZpFHN0G4Qx-XCd/view?usp=sharing`

[26]The x axis is the time base for the output file and each source file's y axis represents the position from that file, the top of

of converting the 11 dimensional space to a Hamiltonian Path is essentially reducing the space down to 1 dimension, I next decided to use UMAP (Uniform manifold approximation and projection) (McInnes, Healy, and Melville 2018) as another, different strategy to reduce the number of dimensions of the dataset from 11 principle components to 1 dimension. I then similarly reorganized this sequence of 100 millisecond sound slices into a audio-video file.[27] Figure 40 shows the projection's relation to source sound files.[28] Starting from the 11 principle components in both cases (tsp-solver and UMAP) allows for an interesting comparison in the results.[29] Comparing the aesthetic qualities of each algorithm's output offers some possible heuristics for future Hamiltonian Path implementations.

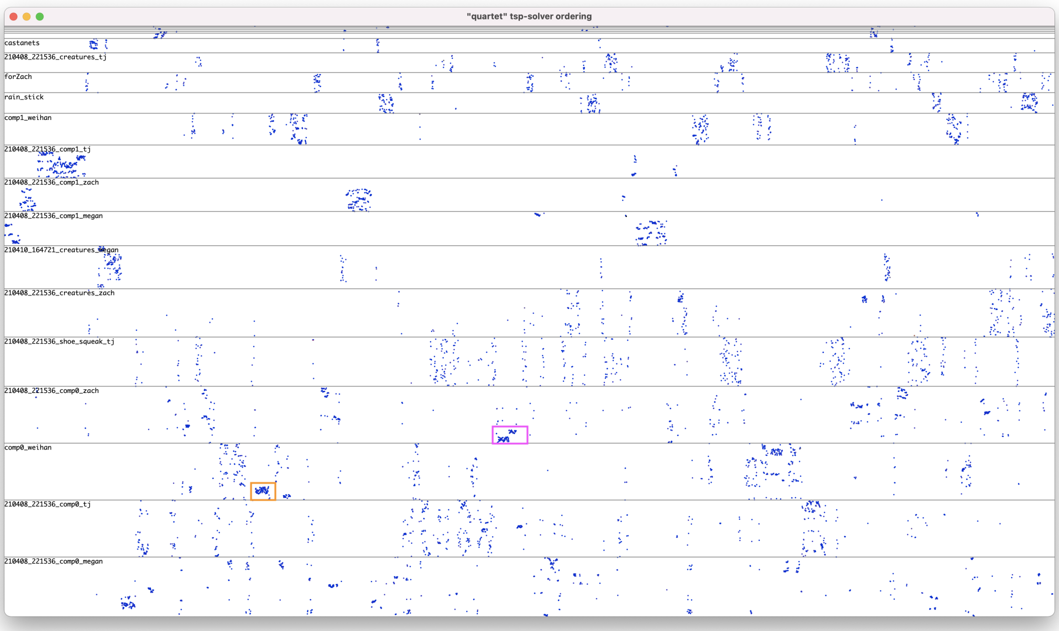


Figure 39: Comparative Matrix of sound source positions in original file (y axis) and positions in minimal distance Hamiltonian Path (Traveling Salesperson Solution) (x axis) for sound sources from "quartet".

The resulting video created with the tsp-solver algorithm (as opposed to UMAP) moves between different timbres more quickly, often giving a sense of gestural transition between them. An initial sequence (from the beginning to about 0:12) moves through a clear clustering of sustained pure sounds from the

the horizonal bar represents the beginning and the bottom represents the end. Color indicates distance to the next point, but most are quite low distances, so color is not particularly valuable here.

[27] `https://drive.google.com/file/d/1chQvJfLZZpXqxU2ao6leAMyMriJRSGyG/view?usp=sharing`

[28] In this plot, color simply represents position in the output soundfile, the same as the x axis.

[29] Also, asking UMAP to reduce a 714 dimensional dataset to 1 dimension would be very CPU intensive and my 11 principle components retain the vast majority of the variance, and therefore remove a lot of redundancy, so it makes sense to use this approach even if comparison was not a goal.

Figure 40: Comparative Matrix of sound source positions in original file (y axis) and positions in ordered one dimensional UMAP projection (x axis) for sound sources from “quartet”.

flute, cello (natural harmonics), and bowed vibraphone, before the path then begins rapidly changing between instruments, not always with clear sonic relationships and connections. These rapid changes create contours in dynamics, pitch, and timbre that imitate gestures or phrases a composer might write. (Of course this may be, in part, because the source material that these 100 millisecond slices are drawn from is recorded audio of musical gestures and phrases that a composer did write.) One such phrase can be heard from 0:21-0:31. In particular, the changing notes in the flute create a melodic contour over these ten seconds, which is also supported by pitches in the cello and piano. Additionally, changes in register, loudness, and timbre give the passage a dynamic energy that I find musically compelling. These quick changes between source files can be seen in Figure 39. Although there is some clear clumping of certain soundfiles near the beginning, the path clearly jumps around somewhat frantically throughout most of the sequence. There are small clumps in various places where one section of one sound file seems to have been focused on intensely for a short period of time.

The video created with the UMAP algorithm has a very different musical sensibility to it. Unlike the tsp-solver video, it does not have abrupt changes in timbre, instead creating longer trajectories of transformation that often cover a more homogenous body of sound. For example from 0:07 to about 0:40

the cello sound transforms from low and forte to high and piano. Other instruments such as the bass drum and flute are mixed into this trajectory in appropriate sounding places. The overall form of this video more clearly divides the instruments into different sections, beginning with the cello trajectory and then moving through large clusters of piano, flute, and percussion. Interestingly, the ending shows a similar clustering as the tsp-solver video beginning, clustering the pure sustained sounds from the flute, cello, and vibraphone. Figure 40 again shows these relationships visually. Unlike Figure 39, one can see the sounfiles more clearly clustered with themselves and even clustered with other soundfiles of similar timbres (such as the "Zach" clustering about half way through which interleaves slices from the files "forZach", "210408_221536_creatures_zach", and "210408_221536_comp0_zach"). The larger trajectors heard in the video can be observed as density crossfades bewtween these clusters such as the opening transitions between source files "210408_221536_comp0_megan" to "210408_221536_comp0_tj" to "210408_221536_shoe_squeak_tj" to "210408_221536_creatures_tj". Comparing Figures 39 and 40, one can also observe similar tight clusters in specific soundfiles in different places in the the timeline (highlighted by colored boxes).

When used in the context of the composition, titled "quartet",[30] these two videos were edited to remove sections I did not like and reorganized to create composed form. Material from the tsp-solver video was mostly used as gestural phrases and connective music, while material from the UMAP video was used for larger formal trajectories.

4 Conclusion: Human-AI Alignment

4.1 A few brief answers to the question "Why?"

After reviewing the four examples of practice-based research in Section 3, one might wonder, Why do all this? Why take on the task of being a data scientist when composing is hard enough? One answer is because I find using these tools to manipulate sound very *exciting*. The conceptutal and sonic ideas these tools offer and the artistic and technical problems they pose stimulate my creative thinking and feed my creative energy in a very valuable way. Using these ideas makes me excited to sit down at my desk and do the work of composition vigorously and often and, I think, that is a good indication of vocation and a good recipe for compelling artistic results.

Another reason is because I find that using new and idiosyncratic processes will lead to new and

[30] `https://vimeo.com/540621361`, password: framerate

idiosyncratic sounds, forms, and compositional conceits–or more generally, new music–thereby expanding, advanding, and individuating my artistic voice. The music that I find most exciting expands or challenges my conception of what music composition is or can be and I try to cultivate that richness of experience in my own work. By expanding and individuating my voice, the experiences that my art offers may become more inventive, thought provoking, and therefore compelling for audiences.[31]

4.2 The Optimization Problem (aka. Composing)

In order to analyze, a little more critically, why a composer might use AI in their compositional process, I propose to think about the act of composition as a classic optimization problem in computer science. Given any synthesis algorithm (whether it be electronic like frequency modulation, or analog such as a violin) (and not to mention the combinatoric possibilities of synthesis algorithms that composers actually use), there is a huge multidimensional space of possible sonic outputs and morphologies. The process of composing necessitates identifying and temporally navigating some smaller subset of these possibilities. This of course is the expression of the composer's agency, the act of composition. Assuming this act carries the intention of creating art that is compelling and that some solutions will be more compelling than others, music composition becomes an optimization problem. How can I, the composer, find the optimal (or near optimal) subset of sonic morphologies (i.e., a compelling music composition) from the given parameter space?

One problem with trying to find the optimal solution is an inaccessibility of some zones within the high-dimensional space of sonic possibilities. Not having access to some of the possible sounds limits the potential for actually acheiving an optimal solution, just like not having access to all roads limits the potential of finding the fastest route home. This inaccessibility may be caused by a lack of technical knowledge, such as not knowing how to notate a woodwind multiphonic and therefore not including it in a composition. Historically, the accessibility of many sonic spaces was pioneered by artists on the avant garde. Only after woodwind players and composers began using multiphonics did that "sonic space" open up as accessible and therefore considerable (by more composers) for inclusion in their optimal solutions. Most often, however, the inaccessibility of some zones is caused by the limitations of the human mind. It is simiply difficult to imagine new combinations of sounds, notes, rhythms, timbres, notations, technologies,

[31] Another reason for using these tools that is less central to my thinking but still worth considering is that, in the new music industry composers are increasingly benefitting from strong individual brands that are centered around salient aspects of their music or artistic practice. (Ritchey 2019) Using AI in my work is also intended to add to my brand as a composer working on the cutting edge of music technology, which can be observed by how I frame my work through prose, but also, hopefully, through the unique sounds and forms my work presents.

playing techniques, interactions, and so many other interconnected dimensions of creativity. Even if it were always *easy* to think of new combinations of parameters, a human lifetime would simply not have the time or energy to actually investigate any considerable size of the possible parameter space.

The analogy of the optimization problem is an oversimplification of the act of composition, namely that I have avoided truly answering what is being optimized, ostensibly the very subjective notion of "the degree of compellingness of the art." However framing this act, in part, as perusing a high-dimensional space for "compelling" sonic forms helps frame why electronic and computer musicians, such as myself, (as well as artists of other disciplines) have pursued employing technological collaborators in their work: the agents created can help increase the breadth and speed of exploring new spaces, as well as surprise the human with unconsidered suggestions of new spaces. The question then becomes, What affordances do different technologies offer to collaborate in this exploration? Which technologies are most useful?

4.3 Human-AI Alignment

One way of measuring which technologies are most useful in approaching the optimization problem is by analyzing which are most able to "absorb" and "understand" the human's musical values, goals, and intentions, thereby being able to interpolate and extrapolate valuable creative suggestions. The "Father of Cybernetics," Norbert Wiener along with Arturo Rosenblueth and Julian Bigelow, say in their landmark paper, "Behavior, purpose and teleology," "All purposeful behavior may be considered to require negative feed-back." (Rosenblueth, Wiener, and Bigelow 1943) Their use of the term negative feedback signifies "that some of the output energy of an apparatus or machine is returned as input;...the behavior of an object is controlled by the margin of error at which the object stands at a given time with reference to a relatively specific goal." When collaborating with a technology to make art, the "relatively specific goal" is the user's artstic goals (whether they be precisely defined or not), therefore the "margin of error" refers the the how closely the system's current output achieves this goal.

The cybernetic feedback loop this creates operates by inviting the technological collaborator to offer a suggestion in the parameter space, which the human can then provide feedback on. This may take the form of indicating that a particular suggetion and everything potentially near it is either interesting or uninteresting (i.e., aligned or unaligned with the human's musical values). Feedback could also be provided by slightly tweaking some suggestion so it becomes more aligned with the human's musical goals. The technology system then has a way of incorporating this feedback so subsequent suggestions may be more aligned with the human's musical goals. Proving this feedback amounts to "training" the system to behave as desired. The more trained and aligned a system becomes with the human's musical values, the more likely it is for the human to perceive mirroring from the system, thereby identifying a collaborative agent. The usefulness of a technology for solving the optimization problem can therefore be measured in how effectively this cybernetic training feedback loop functions, namely the quantity, speed, and quality of information that is relayed in this communication.

In order to analyze the potential for human-AI alignment with different creative technology systems, I analyze below three descriptions of technology training processes in increasing order of potential for alignment: using (1) randomness, (2) complex systems, and (3) machine learning.

4.3.1 Approaching the Optimization Problem with Randomness

Pioneering computation artist Vera Molnár clearly states her use of randomness to approach the optimization problem, saying,

> *There is this old romantic idea which is called "intuition". An artist has talent, a genius, sits down, has a drink and creates. And intuition does what it does. Sometimes it creates something good, sometimes not. Now, when we work with computers we're modern and say intuition is old fashioned. I'm not interested. But, there is a thing that can replace intuition. It's randomness. Because of course, we have more sophisticated machines now it will show you billions of possibilities, of which, with your limited imagination, couldn't have thought of. So it enriches the senses. So, randomness has a lot of importance, but not in the way of Dada. It's not to say "anything can be art". On the contrary, It helps me to better find what I like. Because when you work with intuition, you do ten, twelve, fourteen tests, at the twentyish you're tired and stop. With computers You can first open up the entire spectrum and say, this is the part that interests me and not the rest. So, you place the focus and develop all possibilities within. After you'll find, the interesting part is rather over here, so you get closer.* (Batty 2019)

Clearly she is approaching artistic creation from a computational mindset, using randomness[32] to iterate over possibilities within parameter space and then identifying which results she prefers. She also views the randomness of her computer as an AI collaborator saying it surprises her with things she wouldn't have thought of, yet that she finds artistically interesting. In this section I consider using randomness in this way assumes that parameters are controlled by independent random modulations. If there were extensive interconnections or feedback loop connections I would consider it to be a complex system.

Randomness has also been used as a source of contingency and collaboration in electronic music from its inception. Writing about John Cage's use of radios in *Imaginary Landscape No. 4*, Weinberg states, "Inspired by the Chinese book of oracles, the *I Ching*, Cage demonstrated his fascination with

[32] Although computation random number generators are not truly random, but rather, complex systems, I am treating them here as random, since perceptually this is how they operate.

chance operation, allowing players to control only partial aspects of the composition, while technology, chance, and performers together determined the actual audible content. The role of Cage as a composer was narrowed down to setting the high-level blueprint of dial-setting instructions." (Weinberg 2005) With the development of computer music, new sources of randomness have been created and used. In his essay, *Why do we want our computers to improvise?* George Lewis relates sources of digital randomness[33] in his computer programs to "improvise" saying that he has "made efforts to imbue interactive systems with values such as relative autonomy, apparent subjectivity, and musical uniqueness rather than repeatability." (Lewis 2018)

The problem with randomness One problem with using randomness to explore high dimensional parameter space is that the space is explored, well, randomly. This directionlessness amounts to simply waiting for an interesing combination of parameters to occur, and if this happens, the composer can hit the proverbial save button and have an interesting musical object or pocket of parameter space to explore further. However, waiting makes for an inefficient process of training the technology to come into alignment with the user's goals. One solution, increasing the complexity of the system slightly (by increasing the flow of feedback), is tuning the randomness by selecting weights or distributions (such as Brownian or Guassian) as well as ranges to narrow the randomness's possible output. The feedback process of tuning randomness acts as subtractive filtering, potentially closing off zones within a parameter space that have perhaps yet been underexplored. Furthermore, tuning randomness in this way requires the human to maintain a working knowledge of how each parameter affects the sonic result, which limits the potential for surprise.

Additionaly, using random numbers will not necessarily create a good imitation of human behavior or human improvisation, limiting the potential for them to be percieved as mirroring. Human musical activity (while clearly capable of producing surprise) is not created randomly, but is instead guided by multitudes of prior experiences (training) and influenced by situational contexts (response to stimuli). New Renaissance Artist The Honourable Elizabeth A. Baker explains of human improvisers that, "improvisation is not free and that we need to really stop using the term free improvisation...Humans can't have free improvisation because our mind can only pick from things that it knows". (Baker 2020)

Randomness is categorically unpredictable Technological musical behavior that suggests human-like training and human-like response to stimuli is more likely to be perceived as mirroring one's own (human)

[33]As seen in a diagram included in the essay that shows his use of Max's "drunk" object, which generates "random numbers within a specified step range" (Cycling74 2020)

intentions–more precisely fulfilling Leman's criteria for perceiving agency in music. Leman extends the recognition of intention to also include the ability to predict an agent's actions, saying, "By looking at how a person moves and behaves, I can understand that person as an intentional being. My understanding of his or her intentions allows me to predict his or her actions and understand them as part of an understanding of my own actions," and "Corporeal articulations are conceived as reflecting the action-oriented ontology that is induced by moving sonic forms in music. They exhibit prediction and anticipation of stimulus properties." (Leman 2008) Computer generated randomness is, by design, extremely unpredictable, making it less prone to create actions that can be perceived as mirroring human intentions.

4.3.2 Approaching the Optimization Problem with a Complex System

Randomness and Complexity are Different In pursuing a definition of complexity, Peter Grassberger clearly states that, "complexity is *not* equivalent to randomess, but rather is between order and chaos". (Grassberger 2012) One salient definition of complexity he gives is that "Complex systems are usually composed of many parts, but this alone does not yet qualify them as complex...What is important is that there are strong and non-trivial *correlations* between these parts. Technically, this is best explained via mutual informations". (Grassberger 2012)

While collaborating with a complex system (such as a modular synthesizer in Section 2.2.3), these mutual informations, or non-trivial correlations between parts, are identified as intended correlations between parameters, such as the pitch and filter sweeping together, while the volume crescendos, all in a musical way (that the user would desire, thereby creating mirroring). As described above, if independently randomly modulated, one might have to wait a long time for these parameters to align. If multiple parameters are modulated by the same random source, their correlations would be trivial. By training a complex system, the user can organize and tune the interconnections and feedback loops of the system creating non-trivial correlations between parts. This training orients the system towards making sonic forms more likely or more often aligned with the user's musical goals *and* more likely to create mirroring, all while maintaning a level of complexity that still allows for surprise, as seen in Section 2.2.3.

It should also be observed that the process of creating the patch, prior to the degree of complexity that can induce surprise, is an important part of the training process. It is at this stage that the user embeds in the system artistic choices that already begin to align the technology's musical orientation with their own goals, such as choosing to use frequency modultation rather than subtractive synthesis or whether or not to sample and hold a certain modulation. Each step in this process (perhaps at each added interconnection) offers a sound for the user to then provide feedback on by removing, altering, or adding more interconnections.

Complexity, Mirroring, and Alignment Grassberger's notions of complexity resonate with Leman's concept of mirroring by explaining complexity as phenomenological meaning, stating, "More important than correlations among the parts of a complex system are often correlations between the object and its environment...the letters of a novel in some language could hardly be considered as complex if there were no possibility in principle to read it. The ability to read a novel, and to perceive it thus as complex, requires some correlation between the novel and the reader". (Grassberger 2012) The complexity of an object can exist only in relation to its environment just as the agency of Leman's moving sonic form can exist only in relation to a listener. In both cases these relations need to have some "correlation", some critera upon which the relation is based (e.g. musical simultenaity or shared directionality), and a critera of non-triviality that enable a listener to identify perceived correlations as non-trivial, or in Leman's case, mirrored. In Grassberger's example the criteria is a shared written language of vocabulary and grammar. Musically, recognizing an agent through mirroring or non-trivial correlations is similarly based on a shared vocabulary and grammar of music and sound, one that may be more ubiquitous, such as 18th century counterpoint, or the idiosyncratic voice of a composer.

Grassberger goes on to explain that the *meaning* conveyed through non-trivial correlations is similar to Claude Shannon's notion of information density in a message. The amount of information (in this case musical meaning) a message conveys not only depends on what is in the message but on the context in which it is received (i.e., the legibility to the receiver). The indexical relations it points to are taken on as part of the meaning conveyed. Grassberger states that, "a situation acquires some meaning to us if we realize that only some of its features are essential, that these features are related to something we have already stored in our memory, or that its parts fit together in some unique way. Thus we realize that we can replace a full, detailed and straightforward description by a compressed and more 'intelligent' description which captures only these essential aspects, eventually employing already existing information". (Grassberger 2012)

Similar to the subjective definition of AI in Section 2.1, both Grassberger's definition of complexity and Leman's definition of sonic agency point towards a subjective understanding of human-AI alignment. It is not only left up to the subject to identify what technology is an agent, is complex, and is aligned with their values, but those identifications necessarily rely on the degree to which the AI expresses creative suggestions with a grammar and vocabulary legible to the user. Because in the case of musician-technology collaborations the user is often the person designing the AI collaborator and brining it into alignment through training, this shared language, however idiosyncratic, is central to the collaboration.

4.3.3 Approaching the Optimization Problem with Machine Learning

Written before the recent wave of interest in artificial intelligence, George Lewis observed the reflexive nature of system design and the centrality of a shared vocabulary between user(s) and a system, saying, "Musical computer programs, like any texts, are not 'objective' or 'universal,' but instead represent the particular ideas of their creators. As notions about the nature and function of music become embedded into the structure of software-based musical systems and compositions, interactions with these systems tend to reveal characteristics of the community of thought and culture that produced them." (Lewis 2000) While Lewis describes heuristic choices made by system designers, Rebecca Fiebrink has more recently observed that the act of embedding the "characteristics of the community of thought and culture" can instead now be done with machine learning algorithms through training examples, stating,

> *There are certain things that we care about, as musicians for example, that are really hard to articulate in code [(i.e., heuristically)]. It's hard for me to talk about what kind of quality of sound I want and then translate that into a set of filter coefficients. It's hard for me to talk about how I want a performer to move on stage and then translate that into some sort of mathematical equation for their trajectory. But it's a lot easier for me to either find examples of sounds that have a particular quality or to give examples of movements or if I'm using other types of modalities, often curating or creating examples are just way easier for us as people. And this relates to the types of tacit knowledge and embodied knowledge we bring to creative practices. Even if you're an expert coder, you've had this experience of–there's stuff that's just really hard...to articulate–it's not because you need to be a better coder, it's because as humans, there's stuff that we can breakdown into math easily and there's stuff that we can't* (CeReNeM 2019)

Fiebrink makes clear how machine learning uses a process, different from either randomness or a comlex system, to train the human's musical values and goals into the system, than either randomness or complex system. Rather than waiting for randomness to create mirroring or non-trivial relations, or manually tuning the interconnections of a complex system to bring it into alginment, training using machine learning consists of creating a dataset that expresses the musical values and goals the system should mirror and the non-trivial relationships the user desires. Existing as a dataset, these relationships and values are independent of the machine learning system that will learn them.

Training the system itself then functions as an iterative cybernetic feedback loop (similar to tuning randomness and building up a complex system), however rather than iterating directly with and getting feedback from the human's ears and musical judgement, it instead iterates with the human's musical judgement, getting feedback through the dataset provided. The algorithm that trains a neural network, called back-propogation, works by coming up with possible outputs (randomly at first), then checking the dataset to see to what degree the output aligns with the human's musical values and goals. It measures an error (as described by Wiener, Rosenblueth, and Bigelow) then corrects itself slightly (by tuning the internal interconnections) to try to make a better suggestion next time. These error-correcting iterations happen at the maximum speed of the computer, going through an entire dataset thousands of times before a user reingages directly with the system. This training cybernetic loop between the human's values and the system's output is clearly faster and able provide higher quality information than manually training randomness or a complex system.

The ability to train machine learning systems so quickly allows for an additional cybernetic feedback meta-loop. Users are able to train various datasets, hear how each one might behave and provide feedback to the meta-system about which *trainings* are most aligned with their musical values. While tuning randomness or training a complex system both necessitate closing off zones of the high dimentional parameter space, the meta-loop of training on multiple datasets allows the user to keep the possible parameter space wide open as different datasets and their trainings can access different zones in the high dimensional space. Each dataset used can act as an individually trained complex system. Using machine learning for approaching the optimization problem is incredibly useful as one is able to clearly express musical preferences through datasets, quickly bring the AI system into alignment through backpropagation, and easily switch between trainings, maintaning access to the entire parameter space.

Being surprised by a neural network. Employing a machine learning algorithm as a collaborator requires, not only alignment, but also surprise. The user still needs to experience the outpus of the system as useful creative suggestions they they might not have come up with on their own. In the example of Richard Devine using the complex system of his modular synthesizer as a collaborator (Section 2.2.1), one reason he is able to be surprised is because he is not able to keep track of all the interconnections that are creating the sound. A neural network has hundreds, often thousands, of hidden parameters that the user does not interact with. The inability of the user to even identify, let alone keep track of, these parameters primes the user's experience for creating surprise. When using a neural network (or other machine learning algorithms) there is often a sense of wonder at what it does and creates, a sense of surprise that sufficiently separates its outputs from seeming trivially connected to the inputs.

Conclusion To sum up, these three collaborative technologies all have the potential for to create surprise and mirroring and therefore can be perceived as aligned AI collaborators. Randomness is least likely to offer this result as one must wait for random modulations to combine just right. Through tuning randomness, one can increase this possibility some, but this also requires the intimate working knowledge and maintainence of many parameters, limiting the potential for surprise. Complex systems are more likely to be perceived as AI collaborators because they allow for the tuning of interconnections and feedback loops that more frequently can create non-trivial correlations between parts of the system. Focusing one's attention on these interconnections allows one to loose track of the individual parameter modulations, increaseing the possibility for surprise, while keeping a working knowledge of the system as a whole. Machine learning algorithms greatly increase the speed and precision of training a system, thereby affording the most potential for brining a system into alignment through mirroring. This increased facility allows for a further zooming out from individual parameters to allow approaching the machine learning collaborator as multiple easily accessible systems, all capable of differently navigating the parameter space of possibilities.

4.4 The Sweet Spot of Alignment: Assistants and Collaborators

In the AI Saftey and AI Ethics disciplines, the problem of aligning AI and human values is known as "The Alignment Problem". (Christian 2020) The desireable degree of alignment between human and AI is not always clear. Often we think of examples where we *do* want the AI to be as aligned with humans as possible. For example we don't want to create an AI tasked with solving global warming to wipe out the human race as its optimal solution, we want the AI to value human life. However, we often want our AI to *not* be so aligned with humans that it reproduces human bias and prejudice, as has been seen in predicitve policing AI systems based on human curated data. (Christian 2020) Also there are yet debated ethical thought experiments about with *which* human(s) the AI should be aligned, such as whether the self-driving car should hit and kill two jaywalkers, or avoid doing so by swerving off the cliff and killing the occupant.

Extending the composer's agency Employing AI collaborators in music draws correlate consideration. In each of the three examples above (randomness, complex systems, and machine learning), the training process is an imposition of the composer's musical values into the system they are working with. This process is an extension of the composer's agency into the technology, with the hope that the technology will express back to them their own musical values, a further extention of their agency, which creates the cybernetic loop. In this loop, the technology acts as a filter. It is not able to precisely recreate the composer's intentions back to them; the intentions are necessarily altered in some way. If altered beyond recognition (very low alignment), the filter is too strong, mirroring will not occur, and this technology likely will not feel collaborative. If the filter is too weak, and the intentions of the composer are returned to them precisely (e.g., recording audio from the composer and playing it back), the alignment is too strong, and the technology is not a valuable collaborator. In fact, the technology would not be perceived as an agent at all, as described in Section 2.2.2.

Aligning with Humans Measuring alignment as an indication of collaborative value can also be done for human collaborators. If I am looking to collaborate with an improviser, finding someone who always outputs the same or very similar sounds as I do (high alignment) would not bring much value to the performance, as a solo performance would sound essentially the same. A collaborator who only improvises in the style of 18th century keyboard music (very different from me) would make for a musically disjointed performance, as our vocabularies would limit musical communication and meaningful interaction.[34]

The Sweet Spot Often, approaching the optimization problem with a technological collaborator involves finding the right balance of alignment, the right amount of distortion in the filter.[35] For any given problem, the above strategies (randomness, complexity, and machine learning) offer different proclivities for alignment, all of which may be valuable and pursued at different times for different reasons. Although I resist drawing a linear continuum between AI assistants and AI collaborators, comparing different relations between them reveal different points of alignment. Keeping a technological system relatively unaligned, such as with randomness, allows a broad and open-ended exploration of possibilities. Training a machine learning system to a high degree of alignment (such as in Section 3.2) can perform a complex task with high accuracy (thereby *not* offering surprise), creating an AI assistant. Employing a complex system that exhibits both surprise and mirroring can be an redoubtable agent to improvise alongside. Systems with varying degrees of alignment may act in different roles at different times. Engaging and training a technological system is not always in pursuit of the highest degree of alignment, instead it is in pursuit of the degree of alignment that is appropriate for the artistic task at hand.

[34] Although, the metaphor breaks down at some point here. I would be very curious to experiment with this collaboration.

[35] Many machine learning algorithms offer a way of quantifying the degree to which they align with a human's values (as expressed through data). This measurement is called the "loss" or "error", which is a measure of the difference between the algorithm's output and the desired output as specified in the dataset. A low "error" measurement indicates high alignment. Training or retraining machine learning systems to achieve different levels of error (i.e., alignment) may be a useful strategy for approaching a desireable degree of alignment. However, each dataset, context, and useage will produce different errors that will result in different degrees of perceived alignment, therefore the measure cannot be universally compared.

4.5 Conclusion

Palle Dahlstadt clarifies that although training collaborative technologies can be used to extend human agency into a musical system, current AI technology will never be able to behave as a human would. Describing a performance between improvising pianist Magda Mayas and George Lewis' *Voyager*, he says, "Basically [*Voyager*] reacts to what she's playing but it doesn't have the whole picture of her as a musician or anybody else as a musician, I guess because that's too big...No AI system has that...It's always incomplete...there's so many other things that weigh in." (Karlsruhe 2020) Framing the creative act as an optimization problem is reductive because it suggests there is an optimal solution, which is of course not true. Artistic creation is so complex with so many "things that weigh in", it is an immense challege even for humans, the highest powered processing units ever observed. AI systems are no where near, at least currently, able to engage with creativity at the level of sophistication as humans. The technological collaborators described above can *assist* creativity, both as AI *assistants*, but also as AI *collaborators*, assisting in the exploration of artistic possibilities. The cybernetic relationships I develop with these technologies are not in place of any compositional act, they are themselves a compositional act extending my artistic agency futher and more deeply into the world.

Ge Wang states, "At the end of the day, AI systems are built to help humans. The value of such systems lies not solely in efficiency or correctness, but also in human preference and agency." (Wang 2019) I create these cybernetic loops, in part, to filter my creative values and ideas through technological collaborators as a strategy for exploring creative possibilities. The most important filter in this loop, however, is my own artistic preference. This filter determines what artistic ideas get rejected, pursued, explored, and ultimately shared with an audience.

References

Baker, The Honourable Elizabeth A. 2020. *Talking Free Music with Elizabeth A. Baker*. YouTube. Visited on 11/15/2020. https://www.youtube.com/watch?v=5WjmVLd7XEM.

Batty, Joshua. 2019. *Vera Molnar computer art*. YouTube. Visited on 04/25/2021. https://www.youtube.com/watch?v=BCZNNZGz5YI.

Brandtsegg, Øyvind, Sigurd Saue, and Thom Johansen. 2011. "A Modulation Matrix for Complex Parameter Sets." In *NIME*, 316–319.

Bridson, Robert. 2007. "Fast poisson disk sampling in arbitrary dimensions". *ACM SIGGRAPH 2007 Sketches, SIGGRAPH'07*: 2006. doi:10.1145/1278780.1278807.

CeReNeM. 2019. *Rebecca Fiebrink: Machine Learning as Creative Design Tool.* Centre for Research in New Music. YouTube. Visited on 11/23/2020. https://www.youtube.com/watch?v=Qo6n8MuEgdQ.

Chowning, John M. 1973. "The synthesis of complex audio spectra by means of frequency modulation". *Journal of the audio engineering society* 21 (7): 526–534.

Christian, Brian. 2020. *The Alignment Problem: Machine Learning and Human Values.* WW Norton & Company.

Collins, Harry, and Trevor Pinch. 2006. "On Chance and Contingency". *Public*, no. 33.

Cycling74. 2020. *drunk Reference*. Visited on 11/14/2020. https://docs.cycling74.com/max8/refpages/drunk.

Dodge, Charles, and Thomas A Jerse. 1997. *Computer music: synthesis, composition and performance.* Macmillan Library Reference.

Eck, Douglas, and Juergen Schmidhuber. 2002. "Finding temporal structure in music: Blues improvisation with LSTM recurrent networks". In *Proceedings of the 12th IEEE workshop on neural networks for signal processing*, 747–756. IEEE.

Fiebrink, Rebecca, and Laetitia Sonami. 2020. "Reflections on Eight Years of Instrument Creation with Machine Learning".

Fieldsteel, Eli. 2018. *LightMatrix Sound Test 2018-09-17*. YouTube. Visited on 11/07/2020. https://www.youtube.com/watch?v=gSzCbf8i0EY.

Goulart, Fillipe. 2021. *python-tsp*. Visited on 04/21/2021. https://pypi.org/project/python-tsp/.

Grassberger, Peter. 2012. "Randomness, information, and complexity". *arXiv preprint arXiv:1208.3459*.

Hunt, Andy, and Marcelo M Wanderley. 2002. "Mapping performer parameters to synthesis engines". *Organised sound* 7 (2): 97.

I Dream of Wires. 2013. *Richard Devine: IDOW Extended Interview #8 (Analog Voodoo Effect)*. YouTube. https://www.youtube.com/watch?v=Z7naEUAYDfg.

Kang, Laewoo, Steven J Jackson, and Phoebe Sengers. 2018. "Intermodulation: Improvisation and Collaborative Art Practice for HCI". In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–13.

Karlsruhe, ZKM. 2020. *inSonic 2020: Syntheses - Day 2*. YouTube. Visited on 01/17/2021. https://www.youtube.com/watch?v=sooNxK6oQ4c.

Kotecha, Nikhil, and Paul Young. 2018. "Generating Music using an LSTM Network". arXiv: 1804.07300. http://arxiv.org/abs/1804.07300.

Leman, Marc. 2008. *Embodied music cognition and mediation technology.* Cambridge, MA: MIT press.

Lewis, George E. 2000. "Too Many Notes: Computers, Complexity and Culture in Voyager". *Leonardo Music Journal* 10 (2000): 33–39. ISSN: 0961-1215. doi:10.1162/096112100570585.

— . 2018. “Why do we want our computers to improvise?” Chap. 9 in *The Oxford Handbook of Algorithmic Music*, ed. by Alex McLean and Roger T. Dean, 123–130. Oxford: Oxford University Press.

McInnes, Leland, John Healy, and James Melville. 2018. “Umap: Uniform manifold approximation and projection for dimension reduction”. *arXiv preprint arXiv:1802.03426*.

Nierhaus, Gerhard. 2009. *Algorithmic composition: paradigms of automated music generation*. Springer Science & Business Media.

Peters, Deniz. 2013. “Haptic illusions and imagined agency: Felt resistances in sonic experience”. *Contemporary Music Review* 32 (2-3): 151–164.

Pluta, Sam. 2012. “Laptop Improvisation in Multi-Dimensional Space”. PhD thesis, Columbia University.

Ritchey, Marianna. 2019. *Composing Capital: Classical Music in the Neoliberal Era*. University of Chicago Press.

Rosenblueth, Arturo, Norbert Wiener, and Julian Bigelow. 1943. “Behavior, purpose and teleology”. *Philosophy of science* 10 (1): 18–24.

Rowe, Robert. 1993. *Interactive Music Systems: Machine Listening and Composing*. Cambridge, MA: MIT Press.

Schaeffer, Pierre. 2017. *Treatise on Musical Objects: An Essay across Disciplines*. Vol. 20. Univ of California Press.

Simonite, Tom. 2017. “A ‘Neurographer’ Puts the Art in Artificial Intelligence”. *Wired* (). `https://www.wired.com/story/neurographer-puts-the-art-in-artificial-intelligence/`.

SweetwaterSound. 2014. *Richard Devine Interview - The Sweetwater Minute, Vol. 256*. YouTube. `https://www.youtube.com/watch?v=K_5mb_utKdM`.

Tremblay, Pierre Alexandre, et al. 2019. “From collections to corpora: Exploring sounds through fluid decomposition”. In *International Computer Music Conference and New York City Electroacoustic Music Festival*.

Wang, Ge. 2019. *Humans in the Loop: The Design of Interactive AI Systems*. `https://hai.stanford.edu/blog/humans-loop-design-interactive-ai-systems`.

Weinberg, Gil. 2005. “Interconnected musical networks: Toward a theoretical framework”. *Computer Music Journal* 29 (2): 23–39.

Appendices

The code base used in these projects is more extensive than can fit in the appendicies. Only the most relevant code is provided here.

A Code for SynthMIRNRT

code/SynthMIRNRT.sc

```
SynthMIRNRT {
    //classvar uniqueID = 1000;
    var params;
    var time_counter;
    var pre_wait, post_wait;
    var input_msgs;
    var input_pts;
    var synthDef_to_analyze;
    var osc_actions;
    var save_path;
    var n_features = 53; // in first synth!
    var array_to_csv;
    var analysisfilename;
    var n_frames;
    var analysisfilename_melbands;
    var analysisfilename_chroma;
    var numChans;
    var labels_array;
    var n_active_params;

    *new {
        arg params_, save_path_, synthDef_to_analyze,pre_wait_ = 0.1,post_wait_ = 0.1, audio_path = nil, action = nil, verbose = false, numChans_ = 1, csv_data_points = nil;
        ^super.new.init(params_, save_path_, synthDef_to_analyze,pre_wait_,post_wait_, audio_path, action, verbose, numChans_, csv_data_points);
    }

    makeSynthDefs {
        arg numChans;
        SynthDef(\analysis_log_nrt,{
            arg audioInBus, analysis_buf, t_logger = 0;
            var ogsig = In.ar(audioInBus,numChans);
            var sig = Mix(ogsig) * numChans.reciprocal;
            var fft = FFT(LocalBuf(1024),sig);
            var mfcc = FluidMFCC.kr(sig,40);
            var spec = FluidSpectralShape.kr(sig);
            var pitch = FluidPitch.kr(sig);
            var loudness = FluidLoudness.kr(sig);
            var vector = mfcc ++ spec ++ pitch ++ /*chroma ++*/ loudness ++ [
                A2K.kr(ZeroCrossing.ar(sig)),
                SensoryDissonance.kr(fft)
            ];
            Logger.kr(vector,t_logger,analysis_buf);
            Out.ar(0,ogsig);
        }).writeDefFile;

        SynthDef(\analysis_log_nrt_melbands,{
            arg audioInBus, analysis_buf, t_logger = 0;
            var sig = Mix(In.ar(audioInBus,numChans)) * numChans.reciprocal;
            var melBands = FluidMelBands.kr(sig,40,maxNumBands:40);
            Logger.kr(melBands,t_logger,analysis_buf);
            //Out.ar(0,sig);
```

```
        }).writeDefFile;

        SynthDef(\analysis_log_nrt_chroma,{
            arg audioInBus, analysis_buf, t_logger = 0;
            var sig = Mix(In.ar(audioInBus,numChans)) * numChans.reciprocal;
            var chroma = Chromagram.kr(FFT(LocalBuf(1024),sig),1024);
            Logger.kr(chroma,t_logger,analysis_buf);
            //Out.ar(0,sig);
        }).writeDefFile;
    }

    create_inputs_from_csv {
        arg path, verbose;
        var csv = CSVFileReader.readInterpret(path,true,true); // these should be normalized because it will use the scalars you passed to scale them!!!!!!!!

        if(n_active_params != csv[0].size,{
            "Number of active params is not equal to the number of dimensions in the csv file.".error;
        });

        n_frames = csv.size;

        input_msgs = List.new;
        input_pts = List.new;

        /*
        csv.postln;
        csv.shape.postln;*/

        csv.do({
            arg input_pt;
            var sub_array = List.new;
            var input_pt_sub_array = List.new;
            var input_idx = 0;

            sub_array.addAll([\n_set,1001]);

            params.do({
                arg param, i;
                var name = param[0];
                var val;

                if(param[1].isKindOf(ControlSpec),{
                    var normed_val = input_pt[input_idx];
                    val = param[1].map(normed_val);
                    input_idx = input_idx + 1;
                },{
                    val = param[1];
                });

                sub_array.addAll([name,val]);
                input_pt_sub_array.add(val);
            });

            input_pts.add(input_pt_sub_array);

            input_msgs.add([time_counter,sub_array.asArray]);
            time_counter = time_counter + pre_wait;
            input_msgs.add([time_counter,[\n_set,1000,\t_logger,1]]);
            input_msgs.add([time_counter,[\n_set,1002,\t_logger,1]]);
            input_msgs.add([time_counter,[\n_set,1003,\t_logger,1]]);
            time_counter = time_counter + post_wait;

            if(verbose,{sub_array.postln});
```

```
        });

        ^input_pts.size;
    }

    /*  *nextUniqueID {
    uniqueID = uniqueID + 1;
    ^uniqueID;
    }*/

    init {
        arg params_, save_path_, synthDef_to_analyze,pre_wait_ = 0.1,post_wait_ = 0.1,
            audio_path_ = nil, action = nil, verbose = false, numChans_ = 1, csv_data_points
             = nil;
        var log = ArrayToCSV.open(save_path_+/+"log.csv");
        var synthDef_to_analyze_name;
        //n_features = 40;//88;//92;//104;

        params = params_;
        pre_wait = pre_wait_;
        post_wait = post_wait_;
        save_path = save_path_;
        numChans = numChans_;

        this.makeSynthDefs(numChans);

        if(synthDef_to_analyze.isSymbolWS,{
            synthDef_to_analyze_name = synthDef_to_analyze;
        },{
            synthDef_to_analyze.writeDefFile;
            synthDef_to_analyze_name = synthDef_to_analyze.name;
        });

        time_counter = 0.0;
        analysisfilename = "/tmp/%_nrt_analysis_buf_%.wav".format(Date.localtime.stamp,
            UniqueID.next);
        analysisfilename_melbands = "/tmp/%_nrt_analysis_buf_melbands_%.wav".format(Date.
            localtime.stamp,UniqueID.next);
        analysisfilename_chroma = "/tmp/%_nrt_analysis_buf_chroma_%.wav".format(Date.
            localtime.stamp,UniqueID.next);

        log.writeLine(["SynthDef name: %".format(synthDef_to_analyze_name.asString)]);
        log.writeLine(["pre_wait",pre_wait]);
        log.writeLine(["post_wait",post_wait]);
        log.writeLine(["save_path",save_path]);
        log.writeLine(["audio_path",audio_path_]);

        n_active_params = params.select({arg param; param[1].isKindOf(ControlSpec)}).size;

        if(csv_data_points.isNil,{
            log.writeLine(["name","min","max","warp","step","n_steps_for_analysis"]);
            n_frames = 1;
            params.do({
                arg param;
                var is_active = param[1].isKindOf(ControlSpec);
                if(is_active,{
                    n_frames = n_frames * param[2];
                    log.writeLine([param[0],param[1].minval,param[1].maxval,param[1].warp.
                        class.asString,param[1].step,param[2]]);
                },{
                    log.writeLine([param[0],param[1]]);
                });
            });
            this.create_input_msgs(verbose);
        },{
```

```
            log.writeLine(["params taken from csv file:",csv_data_points]);
            log.writeLine(["name","min","max","warp","step"]);
            params.do({
                arg param;
                var is_active = param[1].isKindOf(ControlSpec);
                if(is_active,{
                    log.writeLine([param[0],param[1].minval,param[1].maxval,param[1].warp.
                        class.asString,param[1].step]);
                },{
                    log.writeLine([param[0],param[1]]);
                });
            });
            this.create_inputs_from_csv(csv_data_points,verbose);
        });

        labels_array = List.new;
        labels_array.addAll(params.collect({
            arg param_array;
            // "param array: %".format(param_array).postln;
            param_array[0].asString;
        }));

        labels_array.addAll(40.collect({
            arg i_;
            "mfcc%".format(i_.asString.padLeft(2,"0"));
        }));

        labels_array.addAll([
            "spec_centroid",
            "spec_spread",
            "spec_skewness",
            "spec_kurtosis",
            "spec_rolloff",
            "spec_flatness",
            "spec_crest",
            "pitch",
            "pitch_confidence",
            "loudness",
            "loudness_truepeak",
            "zero_crossing",
            "sensory_dissonance"
        ]);

        labels_array.addAll(40.collect({
            arg i_;
            "melband%".format(i_.asString.padLeft(2,"0"));
        }));

        labels_array.addAll(12.collect({
            arg i_;
            "chromagram%".format(i_.asString.padLeft(2,"0"));
        }));

        log.writeLine(["labels of columns:"]);
        labels_array.do({
            arg label, i;
            log.writeLine([i,label]);
        });

        log.close;

        osc_actions = [
            [0.0,[\b_alloc,0,n_frames.asInteger,n_features.asInteger]],
            [0.0,[\b_alloc,1,n_frames.asInteger,40]],// mel bands
            [0.0,[\b_alloc,2,n_frames.asInteger,12]],// chroma
```

```
            [0.0,[\s_new, \analysis_log_nrt, 1000, 0, 0, // name, id, addAction, addTarget
                \audioInBus,11, // start args
                \analysis_buf,0
            ]],
            [0.0,[\s_new, \analysis_log_nrt_melbands, 1002, 0, 0, // name, id, addAction, addTarget
                \audioInBus,11, // start args
                \analysis_buf,1
            ]],
            [0.0,[\s_new, \analysis_log_nrt_chroma, 1003, 0, 0, // name, id, addAction, addTarget
                \audioInBus,11, // start args
                \analysis_buf,2
            ]],
            [0.0,[\s_new,synthDef_to_analyze_name,1001,0,0,
                \outBus,11
            ]],
        ];

        osc_actions = osc_actions ++ input_msgs; // insert them all

        //time_counter = time_counter + 1; // i dont think i need this, i'm trying to remove extraneous time so that i can analyze the file...
        osc_actions = osc_actions ++ [
            [time_counter,[\b_write,0,analysisfilename, "WAV", "float"]],
            [time_counter,[\b_write,1,analysisfilename_melbands, "WAV", "float"]],
            [time_counter,[\b_write,2,analysisfilename_chroma, "WAV", "float"]],
            [time_counter,[\c_set, 0, 0]]
        ];

        //osc_actions.dopostln;
        //input_msgs.postln;
        //input_pts.postln;
        this.runAnalysis(audio_path_,action,verbose);
    }

    create_input_msgs {
        arg verbose;
        input_msgs = List.new;
        input_pts = List.new;
        this.create_input_msgs_r(params,0,nil, verbose);
    }

    create_input_msgs_r {
        arg params_, layer = 0, current_frame = nil, verbose;

        /*      "create input msg r:".postln;
        layer.postln;
        current_frame.postln;
        "".postln;*/

        if(layer < params_.size,{
            params_[layer][2].do({
                arg i;
                var i_n = params_[layer][1].map(i.linlin(0,params_[layer][2]-1,0,1));

                if(current_frame.isNil,{
                    current_frame = Array.newClear(params_.size);
                });
                current_frame[layer] = i_n;
                this.create_input_msgs_r(params_,layer + 1, current_frame.copy, verbose);
            });
        },{
            var sub_array = List.new;
            sub_array.addAll([\n_set,1001]);
```

```
            params_.do({
                arg param, j;
                sub_array.addAll([param[0],current_frame[j]]);
            });

            sub_array = sub_array.asArray;

            input_msgs.add([time_counter,sub_array]);
            time_counter = time_counter + pre_wait;
            input_msgs.add([time_counter,[\n_set,1000,\t_logger,1]]);
            input_msgs.add([time_counter,[\n_set,1002,\t_logger,1]]);
            input_msgs.add([time_counter,[\n_set,1003,\t_logger,1]]);
            time_counter = time_counter + post_wait;

            input_pts.add(current_frame);

            if(verbose,{"sub array: %".format(sub_array).postln});
        });
    }

    runAnalysis {
        arg audio_path, action, verbose;
        var out_file_path = "/dev/null";

        if(audio_path.notNil,{
            out_file_path = audio_path;
        });

        //osc_actions.dopostln;

        // "out file path: %".format(out_file_path).postln;

        // "params before nrt: %".format(params).postln;
        Score.recordNRT(
            osc_actions,
            outputFilePath:out_file_path,
            //headerFormat:"wav",
            options:ServerOptions.new.numOutputBusChannels_(numChans),
            //duration:time_counter + 2,
            action:{
                //analysisfilename.postln;
                SoundFile.use(analysisfilename,{
                    arg sf;
                    var array;

                    array_to_csv = ArrayToCSV.open(save_path+/+"analysis.csv");

                    array = FloatArray.newClear(sf.numFrames * sf.numChannels);

                    sf.readData(array);
                    array = array.clump(n_features);

                    // "first sf done".postln;

                    SoundFile.use(analysisfilename_melbands,{
                        arg sf_mb;
                        var array_mb = FloatArray.newClear(sf_mb.numFrames * sf_mb.
                            numChannels);

                        sf_mb.readData(array_mb);
                        array_mb = array_mb.clump(40); // n mel bands;

                        // "second sf done".postln;

```

```
                        SoundFile.use(analysisfilename_chroma,{
                            arg sf_ch;

                            var array_ch = FloatArray.newClear(sf_ch.numFrames * sf_ch.
                                numChannels);

                            sf_ch.readData(array_ch);
                            array_ch = array_ch.clump(12); // chroma

                            // input points
                            // "params: %".format(params).postln;

                            array_to_csv.writeLine(labels_array);

                            /*                         "array: %".format(array).postln;
                            "input points: %".format(input_pts).postln;
                            //"frame: %".format(frame).postln;
                            "array_mb: %".format(array_mb).postln;
                            "array_ch: %".format(array_ch).postln;*/
                            array.do({
                                arg frame, index;
                                var line = input_pts[index] ++ frame ++ array_mb[index] ++
                                    array_ch[index];

                                /*                             index.postln;
                                line.postln;
                                line.size.postln;
                                "".postln;*/

                                array_to_csv.writeLine(line);
                            });

                            array_to_csv.close;

                            // INDICES (you have to add the number of input params to get
                                the right csv index offset):
                            // mfccs 00-39
                            // spec  40-46
                            // pitch 47-48
                            // loudness 49-50
                            // zeroc 51
                            // sensdis 52
                            // mels 53-92
                            // chroma 93-104

                            action.value;
                        });
                    });
                });
            });
    }
}
```

B Code for Section 3.2.1

code/0/NeuralNetwork.sc

```
NeuralNetworkLayer {
    classvar econst = 2.71828;
    var parent, <size, <weights, biases, <>values, activation, previousLayer, <>error;

    *new {
        arg parent, size, previousLayer, activation = "relu";
        ^super.new.init(parent, size, previousLayer, activation);
```

```
    }

    init {
        arg parent_, size_, previousLayer_, activation_ = "relu";
        parent = parent_;
        size = size_;
        activation = activation_;
        previousLayer = previousLayer_;

        values = Matrix.fill(size,1,{0});

        if(previousLayer.notNil,{
            // this is not the input layer
            weights = Matrix.fill(size,previousLayer.size,{1.0.rand});
            biases = Matrix.fill(size,1,{0.5.rand2});
        });
    }

    feedForward {
        values = ((weights * previousLayer.values) + biases).collect({
            arg val;
            this.activationFunc(val);
        });
        ^values;
    }

    activationFunc {
        arg val;
        activation.switch(
            "relu",{
                //"relu val: %".format(val).postln;
                ^max(0,val);
            },
            "sigmoid",{
                ^(1+econst.pow(val * -1)).reciprocal;
            },
            "linear",{
                ^val;
            },
            "tanh",{
                ^tanh(val);
            }
        );
    }

    // https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
    derivativeActivationFunc {
        arg val;
        activation.switch(
            "relu",{
                var return;
                //"d relu val: %".format(val).postln;
                if(val < 0,{return = 0.0},{return = 1.0});
                ^return;
            },
            "sigmoid",{
                ^(val * (1 - val));
            },
            "linear",{
                ^1.0;
            },
            "tanh",{
                ^(1-val.pow(2))
            }
        );
```

```
    }

    backProp {
        var gradient = values.collect({
            arg val, row, col;
            this.derivativeActivationFunc(val) * error.at(row,col) * parent.learningRate;
        });

        weights = weights + (gradient * previousLayer.values.flop);
        biases = biases + gradient;
    }
}

NeuralNetwork {
    var inputSize, layers, <learningRate;

    *new {
        arg inputSize, learningRate = 0.1;
        ^super.new.init(inputSize,learningRate);
    }

    init {
        arg inputSize_, learningRate_ = 0.1;
        inputSize = inputSize_;
        learningRate = learningRate_;

        layers = List.new;
        layers.add(NeuralNetworkLayer(this,inputSize));
    }

    addLayer {
        arg size, activation;
        layers.add(NeuralNetworkLayer(this,size,layers.last,activation));
    }

    feedForward {
        arg in;
        var out;
        layers[0].values_(Matrix.withFlatArray(in.size,1,in));

        layers[1..].do({
            arg layer;
            out = layer.feedForward;
        });

        ^out;
    }

    train1 {
        arg inputs, targets;
        var return_e;
        targets = Matrix.withFlatArray(targets.size,1,targets);

        // calc errors
        layers.last.error = targets - this.feedForward(inputs);
        ((layers.size-2)..1).do({
            arg layerI;
            var layer = layers[layerI];
            layer.error = layers[layerI + 1].weights.flop * layers[layerI + 1].error;
        });

        // back prop
        ((layers.size-1)..1).do({
            arg layerI;
            layers[layerI].backProp;
```

```
        });
        return_e = layers.last.error.flatten;
        //return_e.postln;
        ^(return_e.pow(2).sum / return_e.size);
    }

    train {
        arg trainingData, nEpochs;
        nEpochs.do({
            arg epoch;
            var err = 0;
            trainingData.scramble.do({
                arg trainingPair;
                err = err + this.train1(trainingPair[0],trainingPair[1]);
            });
            "epoch: %".format(epoch).postln;
            "loss: %\n".format(err).postln;
        });
    }

    trainAndTest {
        arg trainingData, trainPercent;
        var trainingN = (trainingData.size * trainPercent).floor.asInteger;
        var trainingSet = trainingData[0..(trainingN-1)];
        var testingSet = trainingData[trainingN..];
        var nCorrect = 0;
        this.train(trainingSet);
        testingSet.do({
            arg testingPair;
            if(this.feedForward(testingPair[0]).maxIndex == testingPair[1].maxIndex,{
                nCorrect = nCorrect + 1;
            });
        });
        ^(nCorrect / testingSet.size);
    }
}
```

code/0/FeedLightMode.sc

```
FeedLightMaster {
    var modes, currentMode, <>running, turnOnWithOnset/*, >onsetFunc = nil*/;

    *new {
        arg modesArray;
        ^super.new.init(modesArray);
    }

    isRunning {
        ^running;
    }

    init {
        arg modesArray_;
        running = true;
        turnOnWithOnset = false;
        modes = modesArray_ ? [];
    }

    addMode {
        arg feedLightMode;
        modes = modes.add(feedLightMode);
    }

    setMode {
        arg m;
```

```
        if(currentMode != m,{
            // we need to change the mappings!
            currentMode = m;
            //"current mode: %".format(currentMode).postln;
        });
    }

    shuffleLightMappings {
        arg m;
        if(running,{
            modes[(m ? currentMode).asInteger].shuffleLightMappings;
        });
    }

    turnOnWithOnset_ {
        arg bool;
        if(bool,{
            running = false;
            turnOnWithOnset = true;
        },{
            running = true;
            turnOnWithOnset = false;
        });
    }

    onsetTrigger {
        if(turnOnWithOnset,{
            //onsetFunc.action(this);
            running = true;
            turnOnWithOnset = false;
        });
    }

    getRGBWM {
        arg light, data, m;
        var gm;

        if(running,{
            gm = modes[m ? currentMode].lightToGroup[light];

            if(gm == \null,{
                ^[0,0,0,0,0];
            },{
                ^gm.getRGBWM(data);
            });
        },{
            ^[0,0,0,0,0];
        });
    }
}

FeedLightMode {
    var <groupedMappings, <lightToGroup, nLights, >bAtLeastOneLight = true, >
        probOfAssigningLight = 0.5;

    *new {
        arg nLights, groupsArray;
        ^super.new.init(nLights,groupsArray);
    }

    init {
        arg nLights_,groupsArray_;
        nLights = nLights_;

        groupedMappings = groupsArray_ ? [];
```

```
        lightToGroup = Dictionary.newFrom(Array.fill(nLights,{arg i; [i,\null]}).flatten);

        this.shuffleLightMappings;
    }

    addGroup {
        arg group;
        groupedMappings = groupedMappings.add(group);
        this.shuffleLightMappings;
        ^this;
    }

    addGroups {
        arg arrayOfGroups;
        arrayOfGroups.do({
            arg group;
            this.addGroup(group);
        });
    }

    shuffleLightMappings {
        var keyValuePairs = Array.fill(nLights,{
            arg i;
            var fill;
            if(probOfAssigningLight.coin,{ // probability of true
                fill = groupedMappings.choose;
            },{
                fill = \null;
            });
            [i,fill];
        });

        if(bAtLeastOneLight,{
            keyValuePairs.choose[1] = groupedMappings.choose;
        });

        lightToGroup = Dictionary.newFrom(keyValuePairs.flatten);
    }
}

FeedLightGroup {
    var <maps,volSpec;

    *new {
        arg mappings;
        ^super.new.init(mappings);
    }

    init {
        arg mappings;
        maps = Dictionary.new;
        mappings.clump(4).do({
            arg arrayOf4;
            this.addMapping(*arrayOf4);
        });
    }

    addMapping {
        arg mirParam, mirSpec, lightParam, lightSpec;
        maps.put(lightParam,(
            mirParam:mirParam,
            mirSpec:mirSpec,
            lightSpec:lightSpec
        ));
```

```
        ^this;
    }

    getRGBWM {
        arg data, fromNormed = false;
        //var color = ColorRGBHSV.newRGB(0,0,0);
        var color = Color(0,0,0);
        var white = 0;
        maps.keysValuesDo({
            arg lightParam, mappingData;
            lightParam.switch(
                \r,{color.red_(this.unmapmap(data,mappingData,fromNormed))},
                \g,{color.green_(this.unmapmap(data,mappingData,fromNormed))},
                \b,{color.blue_(this.unmapmap(data,mappingData,fromNormed))},
                \w,{white = this.unmapmap(data,mappingData,fromNormed)},
                \h,{color.hue_(this.unmapmap(data,mappingData,fromNormed))},
                \s,{color.sat_(this.unmapmap(data,mappingData,fromNormed))},
                \v,{color.val_(this.unmapmap(data,mappingData,fromNormed))}
            );
        });

        //color.postln;

        ^[
            color.red * 255,
            color.green * 255,
            color.blue * 255,
            white * 255,
            color.val * 255
        ];
    }

    myAmpSpec {
        arg amp;
        ^amp.ampdb.linlin(-70,-10,0,1);
    }

    unmapmap {
        arg data, mappingData, fromNormed;
        if(mappingData.mirParam == \constant,{
            ^mappingData.mirSpec;
        },{
            var unmappedMirParam, mappedLightParam;
            var spec = mappingData.mirSpec;
            if(fromNormed,{
                if(spec == \myAmp,{
                    unmappedMirParam = this.myAmpSpec(data.getParam(mappingData.mirParam,
                        true));
                },{
                    unmappedMirParam = data.getParam(mappingData.mirParam,true);
                });
            },{
                if(spec == \myAmp,{
                    unmappedMirParam = this.myAmpSpec(data.getParam(mappingData.mirParam,
                        false));
                },{
                    unmappedMirParam = spec.unmap(data.getParam(mappingData.mirParam,false))
                        ;
                });
            });
            mappedLightParam = mappingData.lightSpec.map(unmappedMirParam);

            /*          "mir param:     %".format(mappingData.mirParam).postln;
            "raw param:     %".format(data[mappingData.mirParam]).postln;
            "mir spec:      %".format(mappingData.mirSpec).postln;
```

```
            "light spec:   %".format(mappingData.lightSpec).postln;
            "mapped param: %\n".format(mappedLightParam).postln;*/

            ^mappedLightParam;
        });
    }
}

// ColorRGBHSV {
//  var <r = 0, <g = 0, <b = 0, <h = 0, <s = 0, <v = 0;
//  // r, g, b, s, and v are 0 to 1
//  // h is 0 to 1
//  *new {
//      ^super.new;
//  }
//
//  *newRGB {
//      arg r, g, b;
//      ^super.new.setRGB(r,g,b);
//  }
//
//  *newHSV {
//      arg h, s, v;
//      ^super.new.setHSV(h,s,v);
//  }
//
//  setRGB {
//      arg r_, g_, b_;
//      r = r_;
//      g = g_;
//      b = b_;
//      # h, s, v = ColorRGBHSV.rgbToHsv(r,g,b);
//  }
//
//  setHSV {
//      arg h_, s_, v_;
//      h = h_;
//      s = s_;
//      v = v_;
//      # r, g, b = ColorRGBHSV.hsvToRgb(h,s,v);
//  }
//
//  r_ {
//      arg r_;
//      this.setRGB(r_,g,b);
//  }
//
//  g_ {
//      arg g_;
//      this.setRGB(r,g_,b);
//  }
//
//  b_ {
//      arg b_;
//  }
//
//  h_ {
//      arg h_;
//  }
//
//  s_ {
//      arg s_;
//  }
//
//  v_ {
```

```
//      arg v_;
//  }
//
//  *rgbToHsv {
//      arg r, g, b;
//      var min, delta, h, s, v;
//      min = [r,g,b].minItem;
//      v = [r,g,b].maxItem;
//      delta = v - min;
//      if(v != 0,{
//          s = delta / v;
//          },{
//              s = 0;
//              h = 0;
//              ^[h,s,v];
//      });
//
//      if(r == v,{
//          h = (g-b) / delta;
//          },{
//              if(g == v,{
//                  h = 2 + ((b-r) / delta);
//                  },{
//                      h = 4 + ((r-g) / delta);
//              });
//      });
//      h = h * 60;
//      if(h < 0,{h = h + 360});
//      h = h / 360;
//      "h,s,v: %".format([h,s,v]).postln;
//      ^[h,s,v];
//  }
//
//  *hsvToRgb {
//      arg h, s, v;
//      var r, g, b, i,f,p,q,t;
//      i = (h * 6).floor;
//      f = (h * 6) - i;
//      p = v * (1 - s);
//      q = v * (1 - (f * s));
//      t = v * (1 - ((1 - f) * s));
//      [i,f,p,q,t].postln;
//      (i % 6).postln;
//      (i % 6).asInteger.switch(
//          0,{
//              r = v;
//              g = t;
//              b = p;
//          },
//          1,{
//              r = q;
//              g = v;
//              b = p;
//          },
//          2,{
//              r = p;
//              g = v;
//              b = t;
//          },
//          3,{
//              r = p;
//              g = q;
//              b = v;
//          },
//          4,{
```

```
//                r = t;
//                g = p;
//                b = v;
//            },
//            5,{
//                r = v;
//                g = p;
//                b = q;
//            }
//        );
//        "r,g,b: %".format([r,g,b]).postln;
//        ^[r,g,b];
//    }
// }
```

code/0/MyDMXLight.sc

```
MyDMXColor {
    var value = 0, task, updateTime, function, kBus, server, <parent;

    *new {
        arg server, updateTime, parent;
        ^super.new.init(server, updateTime,parent);
    }

    init {
        arg server_, updateTime_, parent_;
        server = server_;
        updateTime = updateTime_;
        parent = parent_;

        kBus = Bus.control(server);
    }

    value {
        ^value;
    }

    setValue {
        arg v;
        this.stopTask(v);
        task = Task({
            inf.do({
                value = v;
                updateTime.wait;
            });
        },SystemClock).play;
    }

    setValueFrequently {
        arg v;
        this.stopTask(v);
        //value = v;
    }

    reset {
        arg to = 0;
        task.stop;
        //function.free;
        if(function.isPlaying,{function.free});
        kBus.set(to);
        //server.sync;
        value = to;
    }

```

```
    stopTask {
        arg resetTo = 0;
        parent.parent.pauseOtherControls;
        this.reset(resetTo);
    }

    fadeToValue {
        arg targetVal, fadeTime, curve = 1;
        var n, startVal;

        startVal = value;

        n = (fadeTime / updateTime).floor;

        /*      "update time: %".format(updateTime).postln;
        "start val: %".format(value).postln;
        "target val: %".format(targetVal).postln;
        "fade time: %".format(fadeTime).postln;
        "curve: %".format(curve).postln;
        "n: %".format(n).postln;*/

        if(n == 0,{
            this.setValue(targetVal);
        },{
            this.stopTask(startVal);

            task = Task({
                n.do({
                    arg i;
                    var w;
                    w = i.linlin(0.0,n-1,0.0,1.0);
                    w = pow(w,curve);
                    /*                  "w: %".format(w).postln;
                    "target val: %".format(targetVal).postln;
                    "start val: %".format(startVal).postln;*/
                    value = (targetVal * w) + (startVal * (1-w));
                    // "value: %".format(value).postln;
                    //server.sync;
                    updateTime.wait;
                });
                //"--fade complete".postln;
            },SystemClock).play;
        });
    }

    fadeFromTo {
        arg startVal, targetVal, fadeTime, curve = 1;
        var n;

        n = fadeTime / updateTime;

        /*"update time: %".format(updateTime).postln;
        "start val: %".format(value).postln;
        "target val: %".format(targetVal).postln;
        "fade time: %".format(fadeTime).postln;
        "curve: %".format(curve).postln;
        "n: %".format(n).postln;*/


        this.stopTask(startVal);

        task = Task({
            n.do({
                arg i;
                var w;
```

```
                w = i.linlin(0.0,n-1,0.0,1.0);
                w = pow(w,curve);
                /*                   "w: %".format(w).postln;
                "target val: %".format(targetVal).postln;
                "start val: %".format(startVal).postln;*/
                value = (targetVal * w) + (startVal * (1-w));
                //"value: %".format(value).postln;
                //server.sync;
                updateTime.wait;
            });
            //"--fade complete".postln;
        },SystemClock).play;
    }

    toggle {
        arg onTime, offTime, onValue_;
        var onValue = onValue_ ? 255;
        this.stopTask;
        task = Task({
            inf.do({
                var wt;
                if(value == 0,{
                    // turn it on
                    value = onValue;
                    wt = onTime;
                },{
                    // turn it off
                    value = 0;
                    wt = offTime;
                });
                //server.sync;
                wt.wait;
            });
        },SystemClock).play;
    }

    toggleRandom {
        arg onMiddle = 0.5, onStandardDeviation = 0.4, onValue_ = 255, offMiddle_ = 0.1,
            offStandardDeviation_ = 0.8;
        var offMiddle = offMiddle_ ? onMiddle;
        var offStandardDeviation = offStandardDeviation_ ? onStandardDeviation;
        var onValue = onValue_ ? 255;

        //[onMiddle, onStandardDeviation, onValue, offMiddle, offStandardDeviation].postln;
        //updateTime.postln;

        this.stopTask;
        task = Task({
            inf.do({
                var wt;
                if(value == 0,{
                    // turn it on
                    value = onValue;
                    wt = max(gauss(onMiddle,onStandardDeviation), updateTime);
                },{
                    // turn it off
                    value = 0;
                    wt = max(gauss(offMiddle,offStandardDeviation), updateTime);
                });
                //server.sync;
                wt.wait;
            });
        },SystemClock).play;
    }

```

```
    runLfo {
        arg freq, min = 0, max = 255;
        this.stopTask((min+max) * 0.5);
        task = Task({
            //var utr = updateTime.reciprocal;
            inf.do({
                arg i;
                value = sin(i * updateTime * freq * 2pi).linlin(-1,1,min,max);
                //server.sync;
                //value.postln;
                updateTime.wait;
            });
        },SystemClock).play;
    }

    playFunc {
        arg func;
        this.stopTask;
        /*      "server: ".post; server.postln;
        "func: ".post; func.postln;*/
        //kBus = Bus.control(server);
        function = func.play(outbus:kBus,fadeTime:0);
        NodeWatcher.register(function,true);
        //func.postln;
        //function.postln;
        task = Task({
            inf.do({
                kBus.get({
                    arg v;
                    //v.postln;
                    value = v;
                });
                //server.sync;
                updateTime.wait;
            });
        },SystemClock).play;
        //task.postln;
    }

    listenToBus01 {
        arg bus, min = 0.0, max = 255.0;
        this.stopTask;
        task = Task({
            inf.do({
                bus.get({
                    arg v;
                    //v.postln;
                    value = v.linlin(0.0,1.0,min,max);
                    //value.postln;
                });
                updateTime.wait;
            });
        },SystemClock).play;
    }
}

MyDMXLight {
    var <parent, offset, <lightType, /*nValsPerLight,*/nValsNeeded, dataArray, /*colorIndicies,*/ colors, updateTime, server, adderArray, systemMasterBrightness = 1, masterIndex = nil;//, lightTypes;

    *new {
        arg server, offset, lightType, updateTime, parent;
        ^super.new.init(server, offset, lightType, updateTime, parent);
    }
```

```
	makeDataAndAdderArrays {
		arg nVals;
		nValsNeeded = nVals;
		dataArray = 0.dup(nValsNeeded);
		adderArray = Array.fill(nValsNeeded,{arg i; i});
	}

	systemMasterBrightness_ {
		arg b;
		systemMasterBrightness = b;
	}

	init {
		arg server_, offset_, lightType_, updateTime_, parent_;
		offset = offset_;
		server = server_;
		lightType = lightType_;
		updateTime = updateTime_;
		parent = parent_;


		/*

		lightTypes:

		0: Spotlight IGB-B18
		1: Chauvet 64 RGBA
		2: the color wheel spotlight from "circle"
		3: the uplights from "circle" and "column"
		4: the ones that LUL used at SXSW
		5: the moving head spots that LUL rented for SXSW

		*/

		colors = Dictionary.newFrom([
			\master,MyDMXColor(server,updateTime,this)
		]);

		if(lightType.isInteger,{

			case
			{(lightType == 0) || (lightType == 1) || (lightType == 3) || (lightType == 4) ||
				 (lightType == 5)}{
				colors.put(\r,MyDMXColor(server,updateTime,this));
				colors.put(\g,MyDMXColor(server,updateTime,this));
				colors.put(\b,MyDMXColor(server,updateTime,this));
				colors.put(\w,MyDMXColor(server,updateTime,this));

				//var nValsNeeded = [6,8,4,4,7,10][lightType];

				lightType.switch(
					0,{
						this.makeDataAndAdderArrays(6);
						// spotlight
						// 255 R G B A 0
						dataArray[0] = 255;
						dataArray[1] = colors.at(\r);
						dataArray[2] = colors.at(\g);
						dataArray[3] = colors.at(\b);
						dataArray[4] = colors.at(\w);
						dataArray[5] = 0;
						/*					colorIndicies = Dictionary.newFrom([
						\r,1,
						\g,2,
```

```
                    \b,3,
                    \w,4
                    ]);*/

                    //dataArray[0] = 255;
                    //dataArray[5] = 0;
                },
                1,{
                    // chauvet slim par 64
                    // R G B A 0 0 0 master
                    this.makeDataAndAdderArrays(8);
                    dataArray[0] = colors.at(\r);
                    dataArray[1] = colors.at(\g);
                    dataArray[2] = colors.at(\b);
                    dataArray[3] = colors.at(\w);
                    // 4
                    // 5
                    // 6
                    dataArray[7] = colors.at(\master);

                    masterIndex = 7;
                    /*                    colorIndicies = Dictionary.newFrom([
                    \r,0,
                    \g,1,
                    \b,2,
                    \w,3
                    ]);*/
                },
                3,{
                    // chauvet color dash batten
                    // master R G B nil nil nil nil
                    this.makeDataAndAdderArrays(4);
                    dataArray[0] = colors.at(\master);
                    dataArray[1] = colors.at(\r);
                    dataArray[2] = colors.at(\g);
                    dataArray[3] = colors.at(\b);

                    masterIndex = 0;
                },
                4,{
                    this.makeDataAndAdderArrays(7);
                    // the flood ones LUL rented for SXSW
                    dataArray[0] = colors.at(\r);
                    dataArray[1] = colors.at(\g);
                    dataArray[2] = colors.at(\b);
                    dataArray[3] = 0;
                    dataArray[4] = colors.at(\w); // actually amber
                    dataArray[5] = 0;
                    dataArray[6] = colors.at(\master);

                    masterIndex = 6;
                },
                5,{
                    // the pin ones LUL rented for SXSW
                    this.makeDataAndAdderArrays(10);
                    dataArray[0] = 0;
                    dataArray[1] = 0;
                    dataArray[2] = colors.at(\r);
                    dataArray[3] = colors.at(\g);
                    dataArray[4] = colors.at(\b);
                    dataArray[5] = colors.at(\w);
                    dataArray[6] = 255;
                    dataArray[7] = colors.at(\master);
                    dataArray[8] = 0;
                    dataArray[9] = 0;
```

```
                        masterIndex = 7;
                });
                this.setMasterLevel(255);
            }
            {lightType == 2}{
                // LFS-75DMX
                // master 255 0 colorWheel
                this.makeDataAndAdderArrays(4);

                colors.put(\colorWheel,MyDMXColor(server,updateTime,this));
                dataArray[0] = colors.at(\master);
                dataArray[1] = 255;
                dataArray[3] = colors.at(\colorWheel);

                masterIndex = 0;
            };
        },{
            // light type is not in, check if it is array (it should be)
            if(lightType.isArray,{
                this.makeDataAndAdderArrays(lightType.size);
                lightType.do({
                    arg param, i;
                    if(param.isNumber.not,{
                        var p = MyDMXColor(server,updateTime,this);
                        colors.put(param,p);
                        dataArray[i] = p;
                        if(param == \master,{
                            masterIndex = i;
                        });
                    },{
                        // param is not a symbol so it must be a const
                        dataArray[i] = param;
                    });
                });
            },{
                // light type is not array or int
                Error("Light Type must be int (for preset) or array (for custom dmx channel
                    ordering)").throw;
            })
        });
        ^[this,nValsNeeded];
    }

    getTupleForMockUp {
        var tuple, masterLevel;
        tuple = (
            r:colors.at(\r).value,
            g:colors.at(\g).value,
            b:colors.at(\b).value,
            w:colors.at(\w).value
        );

        masterLevel = (colors.at(\master).value / 255) * systemMasterBrightness;
        tuple.r = (tuple.r * masterLevel).floor;
        tuple.g = (tuple.g * masterLevel).floor;
        tuple.b = (tuple.b * masterLevel).floor;
        tuple.w = (tuple.w * masterLevel).floor;

        ^tuple;
    }

    getOffset {
        ^offset;
    }
```

```
    getAdderArray {
        ^adderArray;
    }

    getDataArray {
        var sendArray;
        //sendArray = 0.dup(nValsPerLight);

        /*sendArray = dataArray.value;*/
        sendArray = dataArray.collect({
            arg color;
            color.value;
        });

        /*      [\r,\g,\b,\w].do({
        arg k; // key is a symbol of the color, value is the MyDMXColor class instance
        var v = colors.at(k);
        sendArray.put(colorIndicies.at(k),v.getValue);
        });*/

        /*      lightType.switch(
        0,{
        // spotlight
        sendArray[0] = 255;
        masterLevel = masterLevel / 255;
        sendArray = (sendArray * [1,masterLevel,masterLevel,masterLevel,masterLevel,1,1,1]).
            round;
        },
        1,{
        // chauvet
        //sendArray = dataArray;
        sendArray[7] = masterLevel;
        });*/

        // do this to control master on the spotlight thing

        if(masterIndex.isNil,{
            var masterLevel = (colors.at(\master).value / 255) * systemMasterBrightness;
            sendArray = (sendArray * [1,masterLevel,masterLevel,masterLevel,masterLevel,1]).
                floor;
        },{
            sendArray[masterIndex] = (sendArray[masterIndex] * systemMasterBrightness).ceil
        });

        ^sendArray;
    }

    setMasterLevel {
        arg m;
        this.setColor(\master,m);
    }

    setR {
        arg r_;
        this.setColor(\r,r_);
    }

    setG {
        arg g_;
        this.setColor(\g,g_);
    }

    setB {
        arg b_;
```

```
        this.setColor(\b,b_);
    }

    setW {
        arg w_;
        this.setColor(\w,w_);
    }

    getCurrentColor {
        ^Color.new255(colors.at(\r).value,colors.at(\r).value,colors.at(\r).value);
    }

    setH {
        arg hue;
        var cc = this.getCurrentColor;
        cc.hue_(hue);
        this.setColors(cc.red,cc.green,cc.blue,0);
    }

    setS {
        arg sat;
        var cc = this.getCurrentColor;
        cc.sat_(sat);
        this.setColors(cc.red,cc.green,cc.blue,0);
    }

    setV {
        arg val;
        var cc = this.getCurrentColor;
        cc.val_(val);
        this.setColors(cc.red,cc.green,cc.blue,0);
    }

    setColors {
        arg r, g, b, w;
        colors.at(\r).setValue(r);
        colors.at(\g).setValue(g);
        colors.at(\b).setValue(b);
        colors.at(\w).setValue(w);
        /*      this.setColor(\r,r);
        this.setColor(\g,g);
        this.setColor(\b,b);
        this.setColor(\w,w);*/
    }

    /*  toggleColor {
    arg color, onTime, offTime;
    }*/

    setColor {
        arg color, value; // color is a symbol
        colors.at(color) !? (_.setValue(value)) ?? {"NO SUCH COLOR".warn};
    }

    setAll {
        arg red, green, blue, white, master;
        //this.setColor(\master,master);
        colors.at(\master).setValue(master);
        /*      this.setColor(\r,red);
        this.setColor(\g,green);
        this.setColor(\b,blue);
        this.setColor(\w,white);*/
        colors.at(\r).setValue(red);
        colors.at(\g).setValue(green);
        colors.at(\b).setValue(blue);
```

```
        colors.at(\w).setValue(white);
    }

    /*  updateFromCue {
    arg cueArray;
    [\r,\g,\b,\w].do({
    arg c;
    colors.at(c).setValue(cueArray[colorIndicies.at(c)]);
    });
    }*/

    blackOut {
        arg fadeTime = 0;
        //"blackout in light called".postln;
        colors.at(\master).fadeToValue(0,fadeTime);
    }

    reset {
        colors.at(\r).reset;
        colors.at(\g).reset;
        colors.at(\b).reset;
        colors.at(\w).reset;
        colors.at(\master).reset;
    }

    stopAnyColorTasks {
        colors.do(_.stopTask);
    }

    getColor {
        arg c;
        var sendC;
        sendC = colors.at(c);
        if(sendC.isNil,{"NO SUCH COLOR".warn});
        ^sendC;
    }
}

MyDMXMaster {
    classvar <>updateTime = 0.033333333333333;
    var server, lights, /*nValsPerLight = 8,*/ /*adderArray,*/ cues, masterTask,
        totalIndiciesUsed, /*<>updateTime,*/ toDMX, currentCueIndex, cueOrder, fadeTimes,
        subTask, dmxControlForPausing, <>addInArray = nil, mockup, <>verbose = false;//,
        autoFollowTimes;

    *new {
        arg server, lightTypes, dmxStartChan = 1, dmxControlForPausing, mockup;
        // light types:
        // 0: Spotlight IGB-B18
        // 1: Chauvet 64 RGBA
        // 2: LFS-75DMX
        // 3: Chauvet Color Dash Batten
        ^super.new.init(server, lightTypes, dmxStartChan, dmxControlForPausing, mockup);
    }

    reset {
        lights.do({
            arg l;
            l.reset;
        });
    }

    systemMasterBrightness_ {
        arg b;
        lights.do(_.systemMasterBrightness_(b));
```

```
    }

    init {
        arg server_, lightTypes_, dmxStartChan = 1, dmxControlForPausing_, mockup_;
        var offset = dmxStartChan - 1;// -1 because we want to start at the 0th index
        server = server_;
        mockup = mockup_;
        //updateTime !? ({updateTime = 30.reciprocal});

        //"dmx update time: %".format(updateTime).postln;

        NodeWatcher(server);

        //adderArray = Array.fill(nValsPerLight,{arg i; i});

        cues = Dictionary.new;
        cueOrder = [];
        currentCueIndex = -1;
        fadeTimes = Dictionary.new;
        //autoFollowTimes = Dictionary.new;

        toDMX = NetAddr("127.0.0.1",6000);

        //offset = 0;
        lights = lightTypes_.collect({
            arg lightType, i;
            var nValsNeeded;
            //[lightType,i].postln;
            var light;
            # light, nValsNeeded = MyDMXLight(server,offset, lightType, updateTime,this);
            offset = offset + nValsNeeded;
            light;
        });

        totalIndiciesUsed = offset;

        masterTask = Task({
            inf.do({
                this.sendData(this.getDataArray);
                //server.sync;
                updateTime.wait;
            });
        },SystemClock).play;

        dmxControlForPausing_ !? ({
            arg dcfp;
            this.setDMXControlForPausing(dmxControlForPausing_);
        });

        lights.do({
            arg light, i;
            "Light %: ".format(i+1).post;
            light.lightType.switch(
                0,{"Spotlight IGB-B18".postln;},
                1,{"Chauvet 64 RGBA".postln;},
                2,{"LFS-75DMX".postln;},
                3,{"Chauvet Color Dash Batten".postln;}
            );
            "    DMX start channel: %\n".format(light.getOffset + 1).postln;
        });
    }

    getLights {
        ^lights;
    }
```

```
    sendData {
        arg dataArray;
        //dataArray.postln;
        if(addInArray.notNil,{
            dataArray = dataArray + addInArray;
        });

        //dataArray = dataArray;

        dataArray = ["/dmxAll512"] ++ dataArray;

        toDMX.sendMsg(*dataArray);

        if(verbose,{dataArray.postln;});

        if(mockup.notNil,{
            mockup.update(this.getTuplesForMockup);
        });
    }

    getTuplesForMockup {
        ^lights.collect({
            arg light;
            light.getTupleForMockUp;
        });
    }

    getDataArray {
        var tempDMXData = 0.dup(totalIndiciesUsed);

        lights.do({
            arg light;
            tempDMXData.putEach(light.getOffset + light.getAdderArray,light.getDataArray);
        });

        ^tempDMXData;
    }

    setLightColor {
        arg light, color, value;
        //this.stopSubTask;
        lights[light].setColor(color,value);
    }

    setLightAll {
        arg light, r, g, b, w, master;
        lights[light].setAll(r,g,b,w,master);
    }

    getLightColor {
        arg light, color;
        ^lights[light].getColor(color);
    }

    getLight {
        arg light;
        ^lights[light];
    }

    postCurrentLook {
        arg cueNumber;
        this.getDataArray.postln;
    }

```

```
    blackOut {
        arg fadeTime = 0;
        //"blackout in master called".postln;
        lights.do({
            arg light;
            light.blackOut(fadeTime);
        })
    }

    setDMXControlForPausing {
        arg dmxControl_;
        dmxControlForPausing = dmxControl_;
        masterTask.pause;
    }

    pauseOtherControls {
        dmxControlForPausing !? (_.pause);
        masterTask.isPlaying.not.if({masterTask.play});
    }

    playOtherControls {
        masterTask.pause;
        dmxControlForPausing !? (_.play);
        //dmxControlForPausing !? (_.play);
    }

    setOtherControlsToPreset {
        arg p;
        this.playOtherControls;
        "---Setting other controls to preset: %".format(p).postln;
        dmxControlForPausing !? (_.goToPresetNumber(p));
    }
}
```

C Code for Section 3.2.2

code/1/nn fm 00 make dataset.scd

```
(
~dir = "/Users/ted/Desktop/SCD/flucoma/nn fm testing/";
File.mkdir(~dir);
~stamp = Date.localtime.stamp;

SynthMIRNRT(

    // 1st argument is an array arrays. each contains
    //           (1) the name of the param (as will be passed to the synth) and
    //           (2) a control spec for how this para should be scaled (from 0-1 to what the
        synth expects)

    [
        [
            \cfreq,
            ControlSpec(20,20000,\exp)
        ],[
            \mfreq,
            ControlSpec(20,20000,\exp)
        ],[
            \index,
            ControlSpec(0,20,\lin)
        ]
    ],

    // 2nd argument is the output location for the csv file
```

```

    "%/%_nn_fm_poisson=37542.csv".format(~dir,~stamp),

    // 3rd argument is the synth that you want to exctract descriptors from - NB: needs to have an "outBus" argument !!!!

    SynthDef(\fm_test_nrt,{
        arg cfreq = 20, mfreq = 20, index = 0, outBus = 0;

        // synth stuff
        var sig = SinOsc.ar(cfreq + SinOsc.ar(mfreq,0,index * mfreq));
        //[cfreq,mfreq,index].poll;

        Out.ar(outBus,sig);
    }),

    // 4th argument is either:
    //        - an integer of how many steps you divide each input dimension by in normalized space (e.g., 5 would sample that dimension
    //          at 0, 0.25, 0.5, 0.75, and 1 and then scale that up by the Control spec) (also 5 with three dimensions would be pow(5,3) = 125 data points
    //        - or a path to a csv file with the (normalized) data points you want to use to do the sampling.

    "/Volumes/Ted's 10TB My Book (June 2020)/PROJECT FILES/machine learning/Sampling/Poisson Sampling/poisson_sampling_n_dims/generated_samples/poisson_sample_set_ndims=3_npoints=37542_r=0.03_k=20_2020-07-16_20-39-56.csv",

    0.5, // 5th argument is pre-wait: duration (in NRT) between setting the input parameters and recording the sample of audio descriptors
    0.1, // 6th argument is post-wait: duration (in NRT) between setting the recording the sample of audio descriptors and setting the next input parameters
    nil, // 7th argument is where to put the "audio file" of data, leaving it nil will use temp dir
    {"============= DONE =============".postln;}, // 8th arg: done action
    false // 9th arg: verbosity
);
)
```

code/1/nn fm 01 pare data.scd

```
(
~csv_data = CSVFileReader.readInterpret("/Users/ted/Desktop/SCD/flucoma/nn fm/200726_01 poisson no median filter/200726_114107_nn_fm_poisson=37542.csv",startRow:1);
~csv_data.size.postln;
~csv_data = ~csv_data.select({
    arg row;
    //cfreq < 7500      mfreq  < 3000      mfreq  < cfreq       ((index  + 1) * mfreq) < cfreq
    (row[0] < 5000) && (row[1] < 2500) && (row[1] < row[0]) && (((row[2] + 1) * row[1]) < row[0])
});

ArrayToCSV(~csv_data,"/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_7500_3k_mfreq<cfreq_indexCalc.csv");
"done";
~csv_data.size.postln;

)

// peek to get a sense of it
(
~keep_indices = [0,1,2] ++ (43..55);
~csv_data = ~csv_data.collect({
    arg row;
```

```
    row.atAll(~keep_indices);
});
~normed_data = MinMaxScaler.fit_transform(~csv_data);
~headers = ["cfreq","mfreq","index","spec_centroid",
    "spec_spread",
    "spec_skewness",
    "spec_kurtosis",
    "spec_rolloff",
    "spec_flatness",
    "spec_crest",
    "pitch",
    "pitch_confidence",
    "loudness",
    "loudness_truepeak",
    "zero_crossing",
    "sensory_dissonance"
];
PlotXYColor(~normed_data,{
    arg idx;
    idx.postln;
    ~csv_data[idx][0..2].postln;
    ~csv_data[idx][3..].postln;
},~headers,slewTime:0);
)
```

code/1/nn fm 02 training.scd

```
(
s.options.device_("Scarlett 6i6 USB");
s.waitForBoot({
    var train = {
        arg fm_json, analysis_json, analysis_name, nSteps, shape;
        Task({
            var timestamp = Date.localtime.stamp;
            //var dir = "/Users/ted/Desktop/SCD/flucoma/nn fm/200724_01/%".format(timestamp);
            var dir = "%%_%_%_shape=%".format(PathName(analysis_json).pathOnly,Date.localtime.stamp,nSteps,analysis_name,shape);
            //var fm_json = "/Users/ted/Desktop/SCD/flucoma/nn fm/200723_01/200718_202553_nn_fm_nSteps=30_fm.json";
            //var analysis_json = "/Users/ted/Desktop/SCD/flucoma/nn fm/200723_01/200718_202553_nn_fm_nSteps=30_entire_analysis.json";
            //var analysis_name = "entire_analysis";

            // read
            var fm = FluidDataSet(s,(\fm++UniqueID.next).asSymbol);
            var fm_norm;
            var fm_norm_ds;
            var analysis;
            var analysis_norm;
            var analysis_norm_ds;
            var hidden_act, output_act, activation_ints, maxIter, net;
            var run_fit;

            "------- dir:           %".format(dir).postln;
            "------- fm json:       %".format(fm_json).postln;
            "------- analysis json: %".format(analysis_json).postln;
            "------- name:          %".format(analysis_name).postln;
            "------- nSteps:        %".format(nSteps).postln;
            "------- shape:         %".format(shape).postln;
            "".postln;

            File.mkdir(dir);

            s.sync;
```

```
            fm.read(fm_json);
            s.sync;
            //fm_stand = FluidStandardize(s);
            //s.sync;
            //fm_stand_ds = FluidDataSet(s,\fm_stand);
            //s.sync;
            //fm_stand.fitTransform(fm,fm_stand_ds,{"done".postln;});
            //s.sync;
            //fm_stand.write("/Users/ted/Desktop/SCD/flucoma/nn fm/200718_01/%
                _fm_stand_nPoints=30.json".format(timestamp));
            //s.sync;
            fm_norm = FluidNormalize(s);
            s.sync;
            fm_norm_ds = FluidDataSet(s,(\fm_norm++UniqueID.next).asSymbol);
            s.sync;
            fm_norm.fitTransform(fm,fm_norm_ds/*,{"done".postln;}*/);
            s.sync;
            fm_norm.write("%/%_fm_norm_nSteps=%.json".format(dir,timestamp,nSteps));
            s.sync;

            // analysis data
            analysis = FluidDataSet(s,(\analysis++UniqueID.next).asSymbol);
            s.sync;
            analysis.read(analysis_json);
            s.sync;
            /*analysis_stand = FluidStandardize(s);
            s.sync;
            analysis_stand_ds = FluidDataSet(s,\analysis_stand);
            s.sync;
            analysis_stand.fitTransform(analysis,analysis_stand_ds,{"done".postln;});
            s.sync;
            analysis_stand.write("/Users/ted/Desktop/SCD/flucoma/nn fm/200718_01/%
                _mfcc_stand_nPoints=30.json".format(timestamp));
            s.sync;*/
            analysis_norm = FluidNormalize(s);
            s.sync;
            analysis_norm_ds = FluidDataSet(s,(\analysis_norm++UniqueID.next).asSymbol);
            s.sync;
            analysis_norm.fitTransform(analysis,analysis_norm_ds/*,{"done".postln;}*/);
            s.sync;
            analysis_norm.write("%/%_%_norm_nSteps=%.json".format(dir,timestamp,
                analysis_name,nSteps));
            s.sync;

            //fm_norm_ds.print;
            //analysis_norm_ds.print;

            // ============================ hyper params =========================
            //shape = [40,30,20,10,5];
            //shape = [3,5,3];
            //shape = [10,6];
            //n = FluidMLPRegressor(s);
            hidden_act = "sigmoid";
            output_act = "identity";
            maxIter = 1000;

            // make network ------------------------------------------------------------
            activation_ints = [hidden_act,output_act].collect({
                arg string;
                var return = nil;
                string.switch(
                    "sigmoid",{return = FluidMLPRegressor.sigmoid},
                    "identity",{return = FluidMLPRegressor.identity},
                    "tanh",{return = FluidMLPRegressor.tanh}
                );
```

```
                return;
            });

            net = FluidMLPRegressor(s,shape,activation_ints[0],activation_ints[1],0,maxIter,0.0001,batchSize:10);
            s.sync;

            run_fit = {
                arg counter;

                net.fit(analysis_norm_ds,fm_norm_ds,{
                    arg error;
                    "".postln;
                    "------- n steps:  %".format(nSteps).postln;
                    "------- analysis: %".format(analysis_name).postln;
                    "------- counter:  %".format(counter).postln;
                    "------- n iters:  %".format(counter * maxIter).postln;
                    "------- shape:    %".format(shape).postln;
                    "------- loss:     %".format(error).postln;
                    "".postln;

                    net.write("%/%_analysis->fm_%_loss=%_nSteps=%_shape=%_hiddenAct=%_outAct=%_nEpochs=%.json".format(
                        dir,
                        timestamp,
                        analysis_name,
                        error.round(0.0001).asString.padRight(6,"0"),
                        nSteps,
                        shape,
                        hidden_act,
                        output_act,
                        counter * maxIter
                    ),{
                        if(error > 0.005,{
                            run_fit.(counter+1);
                        });
                    });
                });
            };

            run_fit.(1);
        }).play;
    };

    /*  ~fm_json = "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_01 poisson no median filter/200726_114107_nn_fm_poisson=37542_5k_2500_mfreq<cfreq_indexCalc_fm.json";
    ~analysis_json = "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_01 poisson no median filter/200726_114107_nn_fm_poisson=37542_5k_2500_mfreq<cfreq_indexCalc_spec.json";*/

    /*  ~large_analysis_json = "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_01 poisson no median filter/200726_114107_nn_fm_poisson=37542_5k_2500_mfreq<cfreq_indexCalc_all_but_chroma.json";*/
    [
        [
            "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_7500_3k_mfreq<cfreq_indexCalc_fm.json",
            "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_7500_3k_mfreq<cfreq_indexCalc_spec.json",
            "spec_filtered_data_7500_3k_indexCalc",5685,[8]
        ]/*,
        [
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_fm.json",
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_spec.json",
```

```
        "spec_filtered_data_10k_5k_indexCalc",6545,[7]
        ],
        [
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_fm.json",
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_spec.json",
        "spec_filtered_data_10k_5k_indexCalc",6545,[6]
        ],
        [
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_fm.json",
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_spec.json",
        "spec_filtered_data_10k_5k_indexCalc",6545,[5]
        ],
        [
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_fm.json",
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_spec.json",
        "spec_filtered_data_10k_5k_indexCalc",6545,[10,5]
        ],
        [
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_fm.json",
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_spec.json",
        "spec_filtered_data_10k_5k_indexCalc",6545,[11,4]
        ],
        [
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_fm.json",
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_spec.json",
        "spec_filtered_data_10k_5k_indexCalc",6545,[9,4]
        ],
        [
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_fm.json",
        "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_02/200726_114107_nn_fm_poisson=37542_10k_5k_mfreq<cfreq_indexCalc_spec.json",
        "spec_filtered_data_10k_5k_indexCalc",6545,[7,4]
        ]*/
    ].do({
        arg arr;
        var fm_json = arr[0];
        var analysis_json = arr[1];
        var analysis_name = arr[2];
        var nSteps = arr[3];
        var shape = arr[4];
        train.(fm_json,analysis_json,analysis_name,nSteps,shape);
    });
});
)
```

code/1/nn fm 03 running.scd

```
(
~render = {
    arg path;
    s.waitForBoot({
        Task({

            var fm_norm_path = "/Users/ted/Desktop/SCD/flucoma/nn fm/200726
```

```
        _172723_fm_norm_nSteps=4584.json";
    var analysis_norm_path = "/Users/ted/Desktop/SCD/flucoma/nn fm/200726
        _172723_spec_filtered_data_norm_nSteps=4584.json";
    var nn_path = "/Users/ted/Desktop/SCD/flucoma/nn fm/200726_172723_melbands->
        fm_spec_filtered_data_nSteps=4584_shape=[ 6 ]_hiddenAct=sigmoid_outAct=
        identity_nEpochs=31800_loss=0.0499.json";

    var analysis_size = 13;

    var test_buf, fm_mins;
    var fm_maxes, analysis_mins, analysis_maxes, fm_ranges, analysis_ranges;
    var fm_json, analysis_json, bus = Bus.audio(s,2);

    s.sync;

    fm_json = JSONFileReader.read(fm_norm_path);
    analysis_json = JSONFileReader.read(analysis_norm_path);

    fm_mins = fm_json.at("data_min").asFloat;
    fm_maxes = fm_json.at("data_max").asFloat;
    fm_ranges = fm_maxes - fm_mins;

    analysis_mins = analysis_json.at("data_min").asFloat;
    analysis_maxes = analysis_json.at("data_max").asFloat;
    analysis_ranges = analysis_maxes - analysis_mins;

    //test_buf = Buffer.readChannel(s,"/Volumes/Ted's 10TB My Book (June 2020)/
        PROJECT FILES/machine learning/Training Data/Audio/a test file.wav",channels
        :[0]);
    //
    //test_buf = Buffer.read(s,"/Users/ted/Documents/_CREATING/_PROJECT FILES/jack/
        sounds (unedited)/SC_200727_171448.aiff");
    //test_buf = Buffer.readChannel(s,"/Volumes/Ted's 10TB My Book (June 2020)/
        PROJECT FILES/machine learning/Training Data/Audio/basson mixed activity for
         testing (complete).wav",channels:[0]);
    //test_buf = Buffer.readChannel(s,"/Volumes/Ted's 10TB My Book (June 2020)/
        PROJECT FILES/machine learning/Training Data/Audio/quick brown fox.wav",
        channels:[0]);
    //test_buf = Buffer.readChannel(s,"/Volumes/Ted's 10TB My Book (June 2020)/SOUND
         DESIGNS/_EURORACK SOUNDS/200613 eurorack 01/_bounces/200613 eurorack 01
        last 10 min excerpt.wav",channels:[0]);

    test_buf = Buffer.readChannel(s,path,channels:[0,1]);
    //test_buf = Buffer.read(s,"/Users/ted/Documents/_CREATING/_PROJECT FILES/wet
        ink/sounds/internal feedback tones STEREO.wav");
    //test_buf = Buffer.read(s,"/Users/ted/Documents/_CREATING/_PROJECT FILES/wet
        ink/sounds/from improv patch/33 filter glitch some glitchy gestures.wav");

    //test_buf = Buffer.readChannel(s,"/Users/ted/Documents/_CREATING/_PROJECT FILES
        /wet ink/sounds 2/slowed down eurorack with chromagram data.wav",channels
        :[0]);

    /*      [fm_mins,fm_maxes,fm_ranges].postln;
    [analysis_mins,analysis_maxes,analysis_ranges].postln;*/

    s.sync;

    ~analysis_channel = {
        arg in_bus, offset = 0, outBus, target;
        Task({
            var net, pitching_bus, catching_bus, input_buf0,input_buf1, output_buf0,
                 trig_rate = 25, analysis_synth, out_synth;
            var amp_bus = Bus.control(s);

            net = FluidMLPRegressor();
```

```
pitching_bus = Bus.control(s);
catching_bus = Bus.control(s);
input_buf0 = Buffer.alloc(s,1,analysis_size);
input_buf1 = Buffer.alloc(s,analysis_size);

output_buf0 = Buffer.alloc(s,3);

s.sync;

net.read(nn_path);

s.sync;

net.synth.moveAfter(target);

s.sync;

net.inBus_(pitching_bus);
net.outBus_(catching_bus);
net.inBuffer_(input_buf1);
net.outBuffer_(output_buf0);

s.sync;

analysis_synth = {
    arg inBus;
    //var stereo = PlayBuf.ar(test_buf.numChannels,test_buf,1,0,rrand(0,test_buf.numFrames),1);
    var sig = In.ar(inBus,1);
    //var sig = stereo
    //var mfcc = FluidMFCC.kr(sig,40)[1..39];
    var spec = FluidSpectralShape.kr(sig);
    var pitch = FluidPitch.kr(sig);
    var loudness = FluidLoudness.kr(sig);
    var zc = A2K.kr(ZeroCrossing.ar(sig));
    var senseDis = SensoryDissonance.kr(FFT(LocalBuf(2048),sig));
    //var melbands = FluidMelBands.kr(sig,maxNumBands:40);
    var trig = Impulse.kr(trig_rate);
    var flat_trig;
    //var vector = mfcc ++ spec ++ pitch;
    //var vector = spec ++ pitch;
    var vector = spec ++ pitch ++ loudness ++ [zc,senseDis];


    Out.kr(amp_bus,DelayN.kr(Amplitude.kr(sig),trig_rate.reciprocal,trig_rate.reciprocal));

    vector = (vector - analysis_mins) / analysis_ranges;

    vector = Median.kr(31,vector);

    RecordBuf.kr(vector,input_buf0);

    flat_trig = FluidBufFlatten.kr(input_buf0,input_buf1,trig:trig);

    Out.kr(pitching_bus,Done.kr(flat_trig));
    //DelayN.ar(Mix(stereo),trig_rate.reciprocal,trig_rate.reciprocal);
    //sig;
}.play(net.synth,args:[\inBus,in_bus.subBus(offset)],addAction:\addBefore);

s.sync;

Buffer.read(s,"/Users/ted/Music/_SAMPLES/eurorack waveforms/VCOb sine.wav",action:{
```

```
                        arg buf;
                        buf.loadToFloatArray(action:{
                            arg float_array;
                            var sized = float_array.resamp1(256);
                            var signal = Signal.newFrom(sized);
                            Task({
                                var wt = signal.asWavetable;
                                var wt_buf = Buffer.loadCollection(s,wt);
                                s.sync;
                                //buf2.plot(name);
                                wt_buf.normalize;
                                s.sync;

                                out_synth = {
                                    var max_del = 8;
                                    var inTrig = In.kr(catching_bus);
                                    var outs = 3.collect({
                                        arg i;
                                        Index.kr(output_buf0,i);
                                    });
                                    var sig;
                                    var cfreq, mfreq, index;
                                    var msig;
                                    var del_time = LFDNoise3.kr(2).range(0,1).pow(2) * max_del;

                                    outs = Median.kr(31,outs);

                                    outs = (outs * fm_ranges) + fm_mins;

                                    outs = outs.lag(trig_rate.reciprocal);

                                    outs = outs ++ [In.kr(amp_bus)];

                                    //outs.poll;

                                    cfreq = outs[0].clip(20,20000);
                                    mfreq = outs[1].clip(20,20000);
                                    index = max(outs[2],0);

                                    //sig = SinOsc.ar(cfreq + SinOsc.ar(mfreq,0,mfreq * index));
                                    msig = Osc.ar(wt_buf,mfreq,3pi/2,mfreq * index);
                                    //sig = Osc.ar(wt_buf,cfreq + msig,3pi/2);
                                    sig = SinOsc.ar(cfreq + msig/*SinOsc.ar(mfreq,0,mfreq * index)*/);

                                    sig = sig * outs[3];
                                    //sig = Pan2.ar(sig,LFDNoise3.kr(FluidLoudness.kr(sig,1,0).linlin(-40,0,0.5,2)));
                                    //Out.ar(0,sig);
                                    Out.ar(outBus,sig);
                                }.play;
                            }).play;
                        });
                    });
                },AppClock).play;
            };

            //s.record;

            ~inSynth = {
                var sig = PlayBuf.ar(2,test_buf,1,0,0,0,2);
                Out.ar(bus,sig);
                //sig;
```

```
            }.play;

            s.sync;

            2.do({
                arg i;
                ~analysis_channel.(bus,i,i,~inSynth);
            });
        }).play;
    });
};
)

(
Task({
    var paths = [
        "/Users/ted/Documents/_CREATING/_PROJECT FILES/barrys album/reaper/barry 01/_bounces
            /stems_200818_143915/01 nim noisy 01.wav",
        "/Users/ted/Documents/_CREATING/_PROJECT FILES/barrys album/reaper/barry 01/_bounces
            /stems_200818_143915/04 nim squishy 01.wav",
        "/Users/ted/Documents/_CREATING/_PROJECT FILES/barrys album/reaper/barry 01/_bounces
            /stems_200818_143915/05 nim squishy 02.wav",
        "/Users/ted/Documents/_CREATING/_PROJECT FILES/barrys album/reaper/barry 01/_bounces
            /stems_200818_143915/06 nim individual pops 01.wav",
        "/Users/ted/Documents/_CREATING/_PROJECT FILES/barrys album/reaper/barry 01/_bounces
            /stems_200818_143915/33 filter glitch some glitchy gestures.wav"
    ];

    paths.do({
        arg path;
        var dur = SoundFile.use(path,{arg sf; sf.duration});

        path.postln;

        s.record;
        ~render.(path);
        dur.wait;
        1.wait;
        s.stopRecording;
        1.wait;
    });

    "================= DONE ===========".postln;

}).play;
)
```

code/1/FMNN 2.sc

```
FM_NN : ImprovModule {
    /* CLASS VARIABLES AND VARIABLES OF ImprovModule CLASS

    classvar <>server, >toLemur;
    var inBus, outBus, group, <cavity, <win, winBounds;

    */
    classvar fm_norm_path = "/Users/ted/Library/Application Support/SuperCollider/Extensions
        /tedsExtensions/machineLearning/NN (basic feedforward)/200726_172723_fm_norm_nSteps
        =4584.json";
    classvar analysis_norm_path = "/Users/ted/Library/Application Support/SuperCollider/
        Extensions/tedsExtensions/machineLearning/NN (basic feedforward)/200726
        _172723_spec_filtered_data_norm_nSteps=4584.json";
    classvar nn_path = "/Users/ted/Library/Application Support/SuperCollider/Extensions/
        tedsExtensions/machineLearning/NN (basic feedforward)/200726_172723_melbands->
        fm_spec_filtered_data_nSteps=4584_shape=[ 6 ]_hiddenAct=sigmoid_outAct=
```

```
        identity_nEpochs=31800_loss=0.0499.json";

    // these variables probably include the actual variables
    // of the module and also variables for each of the GUIs
    var analysis_size = 13;
    var trig_rate = 25;
    var synth;
    var fm_scaler;
    var analysis_scaler;
    var net;

    /* METHODS THAT EACH MODULE MUST HAVE:

    initClass
    init {
    arg inBus_, outBus_, group_, cavity_;
    inBus = inBus_;
    outBus = outBus_;
    group = group_;
    cavity = cavity_;
    }

    free

    inBus_
    outBus_

    pause
    run

    save
    load

    */

    // *initClass {
    //  StartUp.defer {
    //
    //  }
    // }

    init {
        arg inBus_, outBus_, group_, cavity_, onSystemLoad;
        inBus = inBus_;
        outBus = outBus_;
        group = group_;
        cavity = cavity_;

        if(onSystemLoad,{
            this.loadserver;
        },{
            Task({
                this.loadserver;
            }).play(AppClock);
        });
    }

    loadserver {
        analysis_scaler = FluidNormalize(server);
        fm_scaler = FluidNormalize(server);
        net = FluidMLPRegressor(server);

        server.sync;

        analysis_scaler.read(analysis_norm_path);
```

```
        fm_scaler.read(fm_norm_path);
        net.read(nn_path);

        server.sync;

        synth = {
            arg inBus_, trig_rate_ = 25, outBus_, gate = 1, pauseGate = 1;
            var sig = Mix(In.ar(inBus_,4)) * 0.25;
            var spec = FluidSpectralShape.kr(sig);
            var pitch = FluidPitch.kr(sig);
            var loudness = FluidLoudness.kr(sig);
            var zc = A2K.kr(ZeroCrossing.ar(sig));
            var senseDis = SensoryDissonance.kr(FFT(LocalBuf(2048),sig));
            var trig = Impulse.kr(trig_rate_);
            var vector = spec ++ pitch ++ loudness ++ [zc,senseDis];
            var amp = Amplitude.kr(sig);
            var analysis_buf = LocalBuf(analysis_size);
            var analysis_scaled_buf = LocalBuf(analysis_size);
            var nn_out_buf = LocalBuf(3);
            var fm_scaled_buf = LocalBuf(3);
            var outs, cfreq,mfreq, index;

            vector = Median.kr(31,vector);

            vector.do({
                arg val, i;
                BufWr.kr(val,analysis_buf,i,1);
            });

            analysis_scaler.kr(trig,analysis_buf,analysis_scaled_buf);
            net.kr(trig,analysis_scaled_buf,nn_out_buf);
            fm_scaler.kr(trig,nn_out_buf,fm_scaled_buf,invert:1);

            outs = 3.collect({
                arg i;
                BufRd.kr(1,fm_scaled_buf,i,1,1);
            });

            outs = Median.kr(31,outs);

            cfreq = outs[0].clip(20,20000);
            mfreq = outs[1].clip(20,20000);
            index = max(outs[2],0);

            sig = SinOsc.ar(cfreq + SinOsc.ar(mfreq,0,mfreq * index));

            sig = sig * amp;
            sig = sig * EnvGen.kr(Env.asr(0.03,1,0.03),gate,doneAction:2);
            sig = sig * EnvGen.kr(Env.asr(0.03,1,0.03),pauseGate,doneAction:1);
            Out.ar(outBus_,sig.dup(4));
        }.play(group,nil,0,args:[\inBus_,inBus,\trig_rate_,trig_rate,\outBus_,outBus]);
    }

    free {
        Routine{
            this.removeAllAssignments;
            synth.set(\gate,0);

            0.1.wait;

            analysis_scaler.free;
            fm_scaler.free;
            net.free;

            win.close;
```

```
        }.play;
    }

    inBus_ {
        arg inBus_;
        inBus = inBus_;
        synth.set(\inBus,inBus)
    }

    outBus_ {
        arg outBus_;
        outBus = outBus_;
        synth.set(\outBus,outBus);
    }

    pause {
        synth.set(\pauseGate,0);
    }

    run {
        synth.run;
        synth.set(\pauseGate,1);
    }

    /*  save {
    var saves;
    saves = Dictionary.new;

    ^saves;
    }

    load {
    arg saves;
    }*/

    /* OPTIONAL CLASSES FOR INTERFACING WITH LEMUR

    lemurX
    lemurY
    lemurControlPad
    */

    lemurX {
        arg x;
    }

    lemurY {
        arg y;
    }

    lemurControlPad {
        arg cp;
    }
}
```

code/1/MinMaxScaler.sc

```
/*
Ted Moore
www.tedmooremusic.com
ted@tedmooremusic.com
June 4, 2020
*/

MinMaxScaler {
```

```
    var <>ranges;

    initRanges {
        arg size;
        ranges = ControlSpec(inf,-inf).dup(size);
    }

    *fit_transform {
        arg data;
        ^super.new.fit_transform(data);
    }

    *fit {
        arg data;
        ^super.new.fit(data);
    }

    fit {
        arg data;
        this.initRanges(data[0].size);
        //"ranges size: %".format(ranges.size).postln;
        data.do({
            arg entry;
            this.assimilate(entry);
        });
    }

    assimilate {
        arg entry;
        entry.do({
            arg val, i;
            if(val > ranges[i].maxval,{ranges[i].maxval = val});
            if(val < ranges[i].minval,{ranges[i].minval = val});
        });
    }

    transform {
        arg data;
        data = data.collect({
            arg entry;
            entry.collect({
                arg val, i;
                var return = ranges[i].unmap(val);
                if(return.isNaN,{return = 0});
                return;
            });
        });
        ^data;
    }

    fit_transform {
        arg data;
        this.fit(data);
        ^this.transform(data);
    }

    inverse_transform {
        arg data;
        data = data.collect({
            arg entry;
            entry.collect({
                arg val, i;
                ranges[i].map(val);
            });
        });
```

```
        ^data;
    }

    assimilate_transform {
        arg entry;
        var return;
        this.assimilate(entry);
        return = this.transform([entry])[0];
        ^return;
    }
}
```

code/1/PlotXYColor.sc

```
/*
Ted Moore
www.tedmooremusic.com
ted@tedmooremusic.com
June 4, 2020

demo video: https://drive.google.com/file/d/18L7nxhboE3gpEIeuF1etUfJhQ-7uI23u/view?usp=sharing
*/

PlotXYColor {
    var axisFeatureIndex, // dictionary [string of axis -> vector index]
    axisOptions, // array of strings that are the labels of the vector indices (columns)
    axisPums, // pop up menus for selecting what column belongs to what axis
    circleRadius = 6, // how big the dots are
    corpus, // the data that is passed in, but not that data that get's used in the course of things, that's prCorpus
    corpus_dims,
    connector_lines,
    colorArray,
    disp_colors,
    headerArray, // array of strings that user can pass for column headers (OPTIONAL)
    idArray, // array of *anything* (ints, strings, whatever), that the user can pass to be returned on "hover over" (OPTIONAL)
    ignorePrevious, // when the same data point is selected twice in a row, should it be reported twice, or not? (DEFAULT = true)
    lastHovered = nil, // stores the last point that was hovered over
    mouseOverFunc, /* user passed function of what to do when the mouse hovers over a point
    ---------------passed to this function are:
    (0) index of data point (unless idArray is passed in on initialization, in which case the data point's id is passed)
    */
    plotView, // the subview where everything is plotted
    plotWin, // the window
    prCorpus, // a private array of objects that handles the corpus data
    slewTime = 0.5, // how long it takes for the dots to move between different spots in the plot
    filter_index_nb,
    filter_operator_but,
    filter_value_nb,
    justReturnNormXY,
    >blackDot = nil;

    *new {
        arg corpus, mouseOverFunc, headerArray /* optional */, idArray /* optional */, colorArray /* optional */, connector_lines /* optional */, slewTime = 0.5, ignorePrevious = true, justReturnNormXY = false;
        ^super.new.init(corpus,mouseOverFunc,headerArray,idArray,colorArray,connector_lines,slewTime,ignorePrevious,justReturnNormXY);
    }

```

```
/**fromFluidDataSet {
arg ds, mouseOverFunc, headerArray, colorArray, connector_lines, slewTime = 0.5, ignorePrevious = true, action;
Routine{
var norm = FluidNormalize(ds.server);
var norm_ds = FluidDataSet(ds.server);

ds.server.sync;

norm.fitTransform(ds,norm_ds,{
norm_ds.dump({
arg dict;
var data = List.new, ids = List.new;

dict.at("data").keysValuesDo({
arg key, val;
ids.add(key);
data.add(val);
});

data = data.asArray;
ids = ids.asArray;

defer{
action.value(
PlotXYColor(data,mouseOverFunc,headerArray,ids,colorArray,connector_lines,slewTime,ignorePrevious)
);
};
});
});
}.play;
}*/

init {
    arg corpus_, mouseOverFunc_, headerArray_, idArray_, colorArray_, connector_lines_, slewTime_, ignorePrevious_, justReturnNormXY_ = false;
    colorArray = colorArray_;
    connector_lines = connector_lines_;
    corpus = corpus_;
    corpus_dims = corpus[0].size;
    justReturnNormXY = justReturnNormXY_;

    if(corpus_dims < 2,{
        "Corpus must be at least 2 dimensions".throw;
    });

    if((corpus_dims < 3).or(colorArray.notNil),{
        disp_colors = false;
    },{
        disp_colors = true;
    });

    mouseOverFunc = mouseOverFunc_;
    headerArray = headerArray_;
    idArray = idArray_;
    slewTime = slewTime_;
    ignorePrevious = ignorePrevious_;

    // if no header information is passed, make header labels "Feature n"
    if(headerArray.notNil,{
        axisOptions = headerArray;
    },{
        axisOptions = corpus[0].size.collect({
            arg i;
```

```
                "Feature %".format(i);
            });
        });

        this.createPlotWindow;
    }

    createPlotWindow {
        var container;
        plotWin = Window("Plot",Rect(0,0,1200,900))
        .acceptsMouseOver_(true);
        plotWin.view.onResize_({
            plotView.bounds_(Rect(0,20,plotWin.view.bounds.width,plotWin.view.bounds.height-20));
            this.slewDisplay(0);
        });

        // this is just a sub plot for putting the drop down menus in
        container = CompositeView(plotWin,Rect(0,0,plotWin.view.bounds.width,20))
        .background_(Color.white);
        container.decorator_(FlowLayout(container.bounds,0@0,0@0));

        // dictionary lookup (name of axis -> what vector index it is currently displaying)
        axisFeatureIndex = Dictionary.new;

        // make the drop down menus
        axisPums = ["X Axis","Y Axis","Color"].collect({
            arg name, i;
            var pum = nil;

            if(i < corpus_dims,{
                // start with the axis names as displaying columns 0, 1, 2
                axisFeatureIndex.put(name,min(i,corpus_dims-1));

                // make this drop down menu
                StaticText(container,Rect(0,0,50,20)).string_(" " + name);
                pum = PopUpMenu(container,Rect(0,0,160,20))
                .items_(axisOptions) // it has the drop down options made above
                .action_({
                    arg pum;
                    // when something is selected, that index is set in the dictionary to the name of this axis
                    axisFeatureIndex.put(name,pum.value);
                    this.slewDisplay(slewTime); // update the display
                })
                .value_(i); // start it off as 0, 1, or 2 (respectively)
            });

            pum; // return the menu to be part of the axisPums array
        });

        filter_index_nb = EZNumber(container,Rect(0,0,150,20),"Filter Index: ",ControlSpec(0,axisOptions.size-1,step:1),{
            arg nb;
            plotView.refresh;
        },0,false,120,30);

        filter_operator_but = Button(container,Rect(0,0,20,20))
        .states_([[" "],["="],["<"],[">"]])
        .action_({
            arg but;
            plotView.refresh;
        });

        filter_value_nb = EZNumber(container,Rect(0,0,100,20),"Value: ",nil.asSpec,{plotView
```

```
        .refresh;});

    plotView = UserView(plotWin,Rect(0,20,plotWin.view.bounds.width,plotWin.view.bounds.height-20))
    .drawFunc_({ // this is the "draw loop" for a supercollider view - its actually only called though when it needs to be updated
        // i.e. it's not actually looping. this runs everytime plotView.refresh is called.

        prCorpus.do({ // go through the entire private corpus and put a dot on the screen for each
            arg corpusItem, i;
            var draw = this.filterCheck(corpusItem);

            if(draw,{
                Pen.addOval(corpusItem.dispRect);
                if(colorArray.isNil,{
                    if(corpus_dims > 2,{
                        Pen.color_(Color.hsv(corpusItem.color,1,1));
                    },{
                        Pen.color_(Color.black);
                    });
                },{
                    Pen.color_(colorArray[i]);
                });
                Pen.draw;
            });

        });

        if(connector_lines.notNil,{
            //Pen.color_(Color.black);
            //
            connector_lines.do({
                arg pts;
                var pt1 = prCorpus[pts[0]].dispRect.center;
                var pt2 = prCorpus[pts[1]].dispRect.center;
                if(pts.size == 3,{
                    Pen.strokeColor_(pts[2]);
                },{
                    Pen.strokeColor = Color.black;
                });
                /*                    pts.postln;
                pt1.postln;
                pt2.postln;*/
                Pen.line(pt1,pt2);
                Pen.stroke;
            });
        });

        if(blackDot.notNil,{
            Pen.addOval(blackDot);
            Pen.color_(Color.black);
            Pen.draw;
        });

    })
    .mouseOverAction_({ // this function gets called each time the mouse moves over the window
        arg view, px, py, modifiers;
        if(justReturnNormXY.not,{
            var mousePoint = Point(px,py);
            prCorpus.do({ // go through the whole corpus...
                arg corpusItem, i;

```

```
                    if(this.filterCheck(corpusItem),{
                        if(corpusItem.dispRect.notNil,{
                            if(corpusItem.dispRect.contains(mousePoint),{ // if the mouse is inside this datapoint's dot...
                                this.returnIndex(i,px,py); // return the index
                            });
                        });
                    });
                });
            });
        })
        .mouseMoveAction_({ // if the mouse button is down and the mouse moves over the window this function is called
            arg view, x, y, modifiers;
            //["mouse move",view, x, y, modifiers].postln;
            if(justReturnNormXY.not,{
                this.findClosest(x,y); // find the closest point...
            },{
                var nx, ny;
                # nx, ny = this.getrxry(x,y);
                blackDot = Rect(x,y,circleRadius,circleRadius);
                mouseOverFunc.(nx,ny);
            });
        });

        // =============== before we display the window and start using, make the private corpus =============
        prCorpus = corpus.collect({
            arg vector;
            var xindex, yindex, colorindex, dispx, dispy, color;

            // get the vector indicies that are currently assiged to the three axes (here it will obviously be 0, 1, 2)
            # xindex, yindex, colorindex = this.getCurrentIndices;

            // using the axes indices, get the appropriately scaled values for display x pos, display y pos, and display color for this vector
            # dispx, dispy, color = this.getScaledXYColorFromIndices(xindex,yindex,colorindex,vector);

            // each private corpus item has the vector, but also keeps track of where on the screen and what color its dot is
            (vector:vector,dispRect:Rect(dispx,dispy,circleRadius,circleRadius),color:color);
        });

        // update the display stuff
        this.slewDisplay(0);

        // show the window
        plotWin.front;
    }

    setConnectorLines {
        arg cl_arr;
        connector_lines = cl_arr;
        defer{plotView.refresh};
    }

    filterCheck {
        arg corpus_item;
        var draw = true;
        if(filter_operator_but.value != 0,{
            var filter_index = filter_index_nb.value;
            var filter_value = filter_value_nb.value;
```

```
            filter_operator_but.value.switch(
                1,{
                    draw = corpus_item.vector[filter_index] == filter_value;
                },
                2,{
                    draw = corpus_item.vector[filter_index] < filter_value;
                },
                3,{
                    draw = corpus_item.vector[filter_index] > filter_value;
                }
            );
        });
        ^draw;
    }

    getrxry { // pass in an x, y point from the screen (in pixels measurements) and get
        returned the normalized x, y (0 to 1)
        arg px, py;
        var rx = px.linlin(0,plotView.bounds.width,0,1);
        var ry = py.linlin(0,plotView.bounds.height,1,0); // y is inverted for display
            purposes
        ^[rx,ry];
    }

    returnIndex { // this gets called whenever something is going to be passed to the user
        in teh "mouseOverFunc"
        arg idx,px,py; // pass in the index of the data point to be returned and the x, and
            y of that data point in pixels

        // if ignore previous == true, check to make sure this index isn't the most recent
            one. if it is, don't pass it again
        if((idx != lastHovered).or(ignorePrevious.not),{
            var rx, ry, xindex, yindex, colorindex;

            lastHovered = idx; // set "previous" to be this index

            # rx, ry = this.getrxry(px,py); // pass in pixel x,y to get normalized x,y

            # xindex, yindex, colorindex = this.getCurrentIndices; // what are the current
                vector indicies that are being displayed

            // if the user passed in an idArray, don't pass the data point's index, pass the
                 data point's id from that idArray
            if(idArray.notNil,{
                var id = idArray[idx];
                mouseOverFunc.value(id, idx,rx,ry,xindex,yindex);
            },{

                /* evaluate the function the user passed. pass to that function:
                (0) the index (or id) of the point that was hovered over
                (1) the normalized x position of the mouse
                (2) the normalized y position of the mouse
                (3) the current vector index (i.e., feature) that is displayed on the x axis
                (4) the current vector index (i.e., feature) that is displayed on the y axis
                */
                mouseOverFunc.value(idx,rx,ry,xindex,yindex);
            });
        });
    }

    valueActionXY {
        arg x, y, normalized = true;
        if(normalized,{
            x = x.linlin(0,1,0,plotView.bounds.width);
            y = y.linlin(0,1,plotView.bounds.height,0);
```

```
        });
        this.findClosest(x,y);
    }

    findClosest {
        arg x, y;
        var mousePt = Point(x,y);

        var record_dist = inf;
        var winner = nil;
        prCorpus.do({
            arg corpusItem, i;
            if(this.filterCheck(corpusItem),{
                var dist = corpusItem.dispRect.origin.dist(mousePt);
                if(dist < record_dist,{
                    record_dist = dist;
                    winner = i;
                });
            });
        });

        if(winner.notNil,{
            this.returnIndex(winner,x,y);
        });
    }

    getCurrentIndices {
        var xindex = axisFeatureIndex.at("X Axis");
        var yindex = axisFeatureIndex.at("Y Axis");
        var colorindex = axisFeatureIndex.at("Color");
        ^[xindex,yindex,colorindex];
    }

    getScaledXYColorFromIndices { // pass in what vector indices are currently being
        displayed and a vector and get back the appropriately scaled values
        arg xindex, yindex, colorindex, vector;
        var dispx = vector[xindex].linlin(0,1,0,plotView.bounds.width-circleRadius);
        var dispy = vector[yindex].linlin(0,1,plotView.bounds.height-circleRadius,0);
        var color = nil;
        if(colorindex.notNil,{
            color = vector[colorindex].linlin(0,1,0.8,0);// because both 0 and 1 are red...
        });
        ^[dispx,dispy,color];
    }

    slewDisplay {
        arg time = 0.1; // how long should the "slew" take
        time = max(time,0.1);
        Task({
            var startLocs = List.new; // where all the points are starting from (where they
                are right now)
            var endPts = List.new; // where they will be ending up after they slew
            var startColors = List.new; // what color the points are right now
            var endColors = List.new; // what color they will be after the transition
            var updateTime = 30.reciprocal; // reciprocal of the frame rate for the
                animation
            var n_ = time / updateTime; // how many frames of animation will it take to
                complete this transition
            var currentIndices = this.getCurrentIndices; // what are the currently display
                indices (the ones that the user must have just changed to

            prCorpus.do({ // go through each data point
                arg corpusItem;
                var endx, endy, endcolor;
                var color_index = min(2,corpus_dims - 1);
```

```

                startLocs.add(corpusItem.dispRect.copy); // add to this list where this data point is currently (where it will be starting from)
                startColors.add(corpusItem.color); // add to this list what color this data point is currently (where it will be starting from)

                # endx, endy, endcolor = this.getScaledXYColorFromIndices( // get the values that this point will be ending up at
                    currentIndices[0], // x
                    currentIndices[1], // y
                    currentIndices[color_index], // color
                    corpusItem.vector
                );

                endPts.add(Point(endx,endy)); // add to this list where the data point will end its journey
                endColors.add(endcolor); // add to this list the color that the data point will end its journey as
            });

            n_.do({ // do n_ many frames
                arg i;
                var lerp = i.linlin(0,n_-1,-pi,0).cos.linlin(-1,1,0,1); // given i, how far along in the interpolation is the animation
                prCorpus.do({ // go through each corpus item
                    arg corpusItem, i;
                    var ix = lerp.linlin(0,1,startLocs[i].left,endPts[i].x); // given the interpolation amount, what is x
                    var iy = lerp.linlin(0,1,startLocs[i].top,endPts[i].y); // given the interpolation amount, what is y
                    corpusItem.dispRect = Rect(ix,iy,circleRadius,circleRadius); // set this data point's display info to interplation's x,y
                    if(corpus_dims > 2,{
                        corpusItem.color = lerp.linlin(0,1,startColors[i],endColors[i]); // set this data point's color to interpolation color
                    });
                });

                // update display
                plotView.refresh;
                // wait some amount of time before running next animation frame
                updateTime.wait;
            });

        },AppClock).play;
    }
}
```

D Code for Section 3.3.1

code/2/TubeControl.sc

```
TubeControl : ImprovModule {
    /* CLASS VARIABLES AND VARIABLES OF ImprovModule CLASS

    classvar <>server, >toLemur;
    var inBus, outBus, group, <cavity, <win, winBounds;

    */
    // these variables probably include the actual variables
    // of the module and also variables for each of the GUIs
    var synths, inputs, inputSinks, onSystemLoad, useMasterOuts = false, outNBs;//, tubes;

    /* METHODS THAT EACH MODULE MUST HAVE:
```

```

initClass
init {
arg inBus_, outBus_, group_, cavity_;
inBus = inBus_;
outBus = outBus_;
group = group_;
cavity = cavity_;
}

free

inBus_
outBus_

pause
run

save
load

*/

*initClass {
    StartUp.defer {

    }
}

synth {
    ^nil;
}

init {
    arg inBus_, outBus_, group_, cavity_, onSystemLoad_;
    inBus = inBus_;
    outBus = outBus_;
    group = group_;
    cavity = cavity_;
    onSystemLoad = onSystemLoad_;

    synths = nil.dup(3);
    inputs = nil.dup(3);
    inputSinks = nil.dup(3);
    outNBs = nil.dup(3);
    //tubes = ().dup(3);

    this.makeWindow;

    this.addSlider("Q:",ControlSpec(1,10),{
        arg sl;
        synths.do(_.set(\q,sl.value));
    },10,true,false);

    this.addSlider("Lag:",ControlSpec(0.5,5),{
        arg sl;
        synths.do(_.set(\lagTime,sl.value));
    },4,true,false,false);

    /*      Button(win,Rect(0,0,200,20))
    .states_([["IN PARALLEL",Color.white,Color.blue],["IN SERIES",Color.blue,Color.white
        ]])
    .action_({
    arg b;
    if(b.value == 0,{
```

```
        this.parallel;
        },{
        this.series;
        });
        })
        .addToggleRequestNew(
        this.getAddress+/+"series",
        nil,
        this,
        win
        );*/

        this.addSlider("series",nil.asSpec,{
            arg sl;
            synths.do(_.set(\series,sl.value));
        },0,true,true);

        Button(win,Rect(0,0,0,100,20))
        .states_([["Mod Outs"],["Master Outs"]])
        .action_({
            arg but;
            if(but.value == 0,{
                this.setUseMasterOuts(false);
            },{
                this.setUseMasterOuts(true);
            });

        })
        .addToggleRequestNew(
            this.getAddress+/+"masterOuts",
            nil,
            this,
            win
        );

        win.view.decorator.nextLine;

        //*******************************************************
        if(onSystemLoad,{
            this.loadServer;
        },{
            Task({
                //"task played".postln;
                this.loadServer;
            }).play(AppClock);
        });
        //*******************************************************

        //Task({
    }

    setUseMasterOuts {
        arg bool;
        useMasterOuts = bool;
        this.setOuts;
    }

    loadServer {
        [10,8.4166,7.5].do({
            arg feet, i;
            var freqSl,
            partialSlider,
            bpfFreqText,
            anaFreqText,
            anaPitchText,
```

```
            w,
            fundamental,
            color,
            partButs,
            volSl,
            outNB;

            synths[i] = SynthDef(\cm_tubeControl,{
                arg bpff = 880, q = 10, lagTime = 4, random = 0, inBus, vol = 0, outBus, pauseGate = 1, gate = 1, seriesIn, series = 0;
                var in, sig, maxDel = 0.06, freq, hasFreq;
                // this lag time on bpff should really be a Slew.kr !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                bpff = bpff.lag(lagTime);
                //bpff = VarLag.kr(bpff,bpff);
                q = q.lag(0.1);
                in = SelectX.ar(series,[Mix(In.ar(inBus)),Mix(In.ar(seriesIn))]);
                # freq, hasFreq = Pitch.kr(in);
                freq = freq * hasFreq;
                SendReply.kr(Impulse.kr(3),'/anaFreq'++i,freq.lag(0.2));
                random = random.lag(3);
                sig = SelectX.ar(random,[
                    BPF.ar(in,bpff,q.reciprocal),
                    DelayC.ar(in,maxDel,SinOsc.ar(0.01).range(0,maxDel)) // sin freq: 0.01
                ]);
                sig = Compander.ar(sig,sig,-45.dbamp,1,2.reciprocal);
                sig = Compander.ar(sig,sig,-20.dbamp,1,4.reciprocal);
                sig = Limiter.ar(sig,-10.dbamp);
                sig = sig.tanh.softclip(-1,1);
                Out.ar(outBus,sig * vol.dbamp);
            }).play(group,[\inBus,inputs[i],\outBus,outBus.subBus(i)]);//outBus.subBus(i)]);

            color = [Color.cyan,Color.yellow,Color.green][i];
            fundamental = this.metersToFreq(this.feetToMeters(feet)) / 2;
            w = CompositeView(win,Rect(0,0,244,290));
            w.decorator_(FlowLayout(w.bounds));
            w.background_(color);
            StaticText(w,Rect(0,0,100,20)).string_(feet.round(0.1).asString++" FOOT TUBE");
            inputSinks[i] = DragSink(w,Rect(0,0,50,20))
            .action_({
                arg ds;
                this.assignInputBus(i,ds);
            });

            if(inputs[i].notNil,{
                inputSinks[i].object_(inputs[i]);
                inputSinks[i].doAction;
            });

            w.decorator.nextLine;

            server.sync;

            freqSl = EZSlider(w,Rect(0,0,200,20),"BPF",ControlSpec(50,750,\exp),{
                arg sl;
                synths[i].set(\bpff,sl.value);
                bpfFreqText.string_("BPF Pitch:         "++this.freqToPitchAndCents(sl.value));
            },fundamental);

            freqSl.addHandleRequestNew(
                this.getAddress+/+"freq"++i,
                freqSl.controlSpec,
                nil,
                this,
```

```
    w
);

bpfFreqText = StaticText(w,Rect(0,0,200,20));
freqSl.doAction;
anaFreqText = StaticText(w,Rect(0,0,200,20));
anaPitchText = StaticText(w,Rect(0,0,200,20));
w.decorator.nextLine;

Button(w,Rect(0,0,200,20))
.states_([["Play Specified Partial"],["Play Randomly",Color.white,Color.red]])
.action_({
    arg b;
    synths[i].set(\random,b.value);
})
.addToggleRequestNew(
    this.getAddress+/+"random"++i,
    nil,
    this,
    w
);

w.decorator.nextLine;

partButs = 13.collect({
    arg partial;
    var freq, button;
    partial = partial + 1;
    freq = partial * fundamental;
    button = Button(w,Rect(0,0,20,20))
    .states_([[partial,Color.white,Color.black],
        [partial,Color.black,color]
    ])
    .action_({
        arg but;
        //"button value".postln;
        //but.value.postln;
        but.value_(1);
        defer{freqSl.valueAction_(freq)};
        partButs.do({
            arg b;
            if(b !== but,{
                /*"this is not the same button".postln;
                b.value.postln;*/
                if(b.value != 0,{
                    //"this button is not 0".postln;
                    b.value_(0)
                });
            });
        });
        // ******************************************************
        //win.bounds.postln;
        // ******************************************************
    })
    .addToggleRequestNew(
        this.getAddress+/+"tube"++i++"partial"++partial,
        nil,
        this,
        w
    );
    if(partial == 1,{button.doAction});
    button;
});

partialSlider = EZSlider(w,Rect(0,0,200,20),"Partial:",ControlSpec(1,10,\lin,1)
```

```
                ,{
                arg sl;
                partButs[sl.value-1].valueAction_(1);
            },1,true);
            partialSlider.addHandleRequestNew(
                this.getAddress+/+"partialSlider"++i,
                partialSlider.controlSpec,
                nil,
                this,
                w
            );

            volSl = EZSlider(w,Rect(0,0,200,20),"Vol:",\db.asSpec,{
                arg sl;
                synths[i].set(\vol,sl.value);
            },0,false);

            volSl.addHandleRequestNew(
                this.getAddress+/+"vol"++i,
                volSl.controlSpec,
                nil,
                this,
                w
            );

            outNB = EZNumber(w,Rect(0,0,200,20),"Out:",ControlSpec(0,cavityMatrix.
                nOutChannels-1,step:1),{
                arg nb;
                //tubes[i].outChan = nb.value;
                if(useMasterOuts.not && (nb.value < outBus.numChannels).not,{
                    // set to max
                    nb.valueAction_(outBus.numChannels-1);
                },{
                    this.setOuts;
                });
            },i);
            outNB.addHandleRequestNew(
                this.getAddress+/+"out"++i,
                outNB.controlSpec,
                nil,
                this,
                w
            );

            outNBs[i] = outNB;

            //tubes[i].outNB = outNB;

            OSCdef(\anaFreq++i,{
                arg msg;
                var freq;
                freq = msg[3].round(0.1);
                defer{
                    if(freq > 0,{
                        anaFreqText.string_( "Analysis Freq:  "++freq);
                        anaPitchText.string_("Analysis Pitch: "++this.freqToPitchAndCents(
                            freq));
                    },{
                        anaFreqText.string_( "Analysis Freq:  NONE");
                        anaPitchText.string_("Analysis Pitch: NONE");
                    });
                };
            },'/anaFreq'++i);
        });
        this.adjustWindowAndFront;
```

```
        //}).play(AppClock);
    }

    setOuts {
        3.do({
            arg i;
            var bus = outNBs[i].value;
            if(useMasterOuts,{
                synths[i].set(\outBus,bus);
            },{
                synths[i].set(\outBus,outBus.subBus(bus));
            });
        });
    }

    /*  parallel {
    3.do({
    arg i;
    synths[i].set(\inBus,inputs[i]);
    });
    }

    series {
    [2,0,1].do({
    arg input, i;
    synths[i].set(\inBus,inputs[input]);
    });
    }*/

    freqToPitchAndCents {
        arg freq;
        ^(freq.cpsname+this.centsOff(freq)+"cents");
    }

    centsOff {
        arg freq;
        var freqs = (0..127).midicps;
        var name = (freqs[(freqs-freq).abs.minIndex]).cpsname;
        var closestFreq = name.namecps;
        var cents = (freq/closestFreq).log2 * 1200;
        cents = cents.round(1);
        if(cents > 0,{cents = "+"++cents.asString},{cents = cents.asString});
        ^cents;
    }

    metersToFreq {
        arg meters;
        var speed = 340.29; // m/s
        var freq = speed / meters;
        /*"".postln;
        meters.post; " meters =>".postln;
        freq.round(0.01).post; " Hz".postln;
        freq.cpsname.post; " ".post; (this.centsOff.(freq)+"cents").postln;
        "".postln;*/
        ^freq;
    }

    feetToMeters {
        arg feet;
        ^(feet * 0.3048);
    }

    nameToMeters {
        arg name;
        var speed = 340.29; // m/s
```

```
        var freq = name.namecps;
        var meters = speed / freq;
        ^meters.round(0.0001);
    }

    free {
        this.removeAllAssignments;
        synths.do(_.set(\gate,0));
        win.close;
    }

    inBus_ {
        arg inBus_;
        inBus = inBus_;
        //synth.set(\inBus,inBus)
    }

    outBus_ {
        arg outBus_;
        outBus = outBus_;
        this.setOuts;
        /*      synths.do({
        arg sy, i;
        //sy.postln;
        //outBus.subBus(i).postln;
        sy.set(\outBus,outBus.subBus(i));
        });*/
    }

    pause {
        synths.do(_.set(\pauseGate,0));
    }

    run {
        if(synths.size > 0,{
            synths.do({
                arg sy;
                if(sy.notNil,{sy.run; sy.set(\pauseGate,1)});
            });
        });
    }

    save {
        var dict;

        dict = super.save;
        3.do({
            arg i;
            dict.put(\input++i,inputs[i]);
        });

        /*      dict.put(\outChans,
        tubes.collect({
        arg t;
        t.outNB.value;
        });
        );*/

        ^dict;
    }

    load {
        arg dict;
        //Task({
        //1.wait;
```

```
        3.do({
            arg i;
            //("input"+i+"bus"+dict.at(\input++i)).postln;
            dict.at(\input++i) !? ({
                arg input;
                inputs[i] = input;
                if(inputSinks[i].notNil,{
                    inputSinks[i].object_(inputs[i]);
                    inputSinks[i].doAction;
                });
            });
        });

        super.load(dict);

        /*      dict.at(\outChans) !? ({
        3.do({
        arg i;
        tubes[i].outNB.valueAction_(dict.at(\outChans)[i]);
        });
        });*/
        //}).play(AppClock);
    }

    assignInputBus {
        arg i, ds;
        //("input sink"+i+"action done").postln;
        inputs[i] = ds.object;
        synths[i].set(\inBus,inputs[i]);
        synths[(i+1) % 3].set(\seriesIn,inputs[i]);
        if(ds.object.notNil,{
            ds.string_(InputBusAssign.getBusName(ds.object));
        });
    }

    /* OPTIONAL CLASSES FOR INTERFACING WITH LEMUR

    lemurX
    lemurY
    lemurControlPad
    */

    lemurX {
        arg x;
    }

    lemurY {
        arg y;
    }

    lemurControlPad {
        arg cp;
    }
}
```

code/2/feedback control.sc

```
FdbkControl : ImprovModule {
    /* CLASS VARIABLES AND VARIABLES OF ImprovModule CLASS

    classvar <>server, >toLemur;
    var inBus, outBus, group, <cavity, <win, winBounds;

    */
    var privateBus,filterGroup,inSynth,outSynth,threshold,mySpectrogram,magsTask,dict,
```

```
		waitMin, waitMax, controlling, dbDown, fadeTime, updateWaitTime;

	/* METHODS THAT EACH MODULE MUST HAVE:

	initClass
	init {
	arg inBus_, outBus_, group_, cavity_;
	inBus = inBus_;
	outBus = outBus_;
	group = group_;
	cavity = cavity_;
	}

	free

	inBus_
	outBus_

	pause
	run

	save
	load

	*/

	*initClass {
		StartUp.defer {
			SynthDef(\cm_feedbackFilterer_in,{
				arg privateBus,inBus, hpfreq = 240, lpfreq = 1200;
				var /*env,*/ sig; // hpfreq was at 240
				// ****************
				sig = Mix(In.ar(inBus,4));
				//sig = SoundIn.ar(1);
				// ****************
				sig = HPF.ar(HPF.ar(HPF.ar(sig,hpfreq),hpfreq),hpfreq);
				sig = LPF.ar(LPF.ar(LPF.ar(sig,lpfreq),lpfreq),lpfreq);
				//sig = sig * env.dbamp;
				sig = Limiter.ar(sig);
				Out.ar(privateBus,sig);
			}).writeDefFile;

			SynthDef(\cm_feedbackFilterer_notch,{
				arg freq,outBus,waitTime, dbDown, fadeTime;
				var in, dB;
				in = In.ar(outBus);
				//dB = Line.kr(0,-2,4);
				dB = EnvGen.kr(Env([0,dbDown,dbDown,0],[fadeTime,waitTime-fadeTime,2]),
					doneAction:2);
				in = BPeakEQ.ar(in,freq,0.05,dB);
				ReplaceOut.ar(outBus,in);
			}).writeDefFile;

			SynthDef(\cm_feedbackFilterer_out,{
				arg outBus,privateBus,gate = 1,pauseGate = 1;
				var sig, maxDelay = 0.2;
				sig = In.ar(privateBus,1);
				sig = sig.dup(4);
				sig = Limiter.ar(sig);
				sig = sig * EnvGen.kr(Env.asr(0.03,1,0.03),gate,doneAction:2);
				sig = sig * EnvGen.kr(Env.asr(0.03,1,0.03),pauseGate,doneAction:1);
				Out.ar(outBus,sig);
			}).writeDefFile;
		}
	}
```

```

	synth {
		^nil;
	}

	init {
		arg inBus_, outBus_, group_, cavity_;
		inBus = inBus_;
		outBus = outBus_;
		group = group_;
		cavity = cavity_;

		threshold = 5;
		privateBus = Bus.audio(server,1);
		filterGroup = Group(group);
		dict = Dictionary.new;
		controlling = true;
		dbDown = -2;
		fadeTime = 4;

		inSynth = Synth(\cm_feedbackFilterer_in,[\privateBus,privateBus,\inBus,inBus],
			filterGroup,\addBefore);

		//Task({
		//2.wait;
		mySpectrogram = MySpectrogram(privateBus,nil,nil,inSynth,\addAfter,false);
		//2.wait;

		outSynth = Synth(\cm_feedbackFilterer_out,[\outBus,outBus,\privateBus,privateBus],
			filterGroup,\addAfter);

		//2.wait;

		magsTask = Task({
			inf.do({
				arg i;
				//i.postln;
				if(mySpectrogram.magsArray.notNil && controlling,{
					var v = mySpectrogram.magsArray;
					v = v + 1;
					if(mySpectrogram.magsArray.maxItem > threshold,{
						var f = mySpectrogram.freqBuf.get(
							mySpectrogram.magsArray.indexOf(
								mySpectrogram.magsArray.maxItem
							),{
								arg v;
								//v.postln;
								if(dict.keys.includes(v).not,{
									var waitTime;
									waitTime = rrand(waitMin,waitMax);
									//v.postln;
									//dict.keys.size.postln;
									dict.put(v,
										Synth(\cm_feedbackFilterer_notch,[
											\freq,v,
											\outBus,privateBus,
											\waitTime,waitTime,
											\dbDown,dbDown,
											\fadeTime,fadeTime
										],filterGroup);
									);
									AppClock.sched(waitTime,{
										dict.removeAt(v);
										nil;
									});
```

```
                            });

                        });
                    });
                });
                updateWaitTime.wait;
            });
        },AppClock);

        magsTask.play;

        //}).play(AppClock);

        this.makeWindow;

        Button(win,Rect(0,0,180,20))
        .states_([["Controlling",Color.black,Color.green],["Not Controlling",Color.black,
            Color.yellow]])
        .action_({
            arg b;
            if(b.value == 0,{
                controlling = true;
            },{
                controlling = false;
            });
        })
        .addToggleRequestNew(
            this.getAddress+/+"controlling",
            nil,
            this,
            win
        );

        win.view.decorator.nextLine;

        this.addSlider("hpfreq",\freq.asSpec,{
            arg sl;
            inSynth.set(\hpfreq,sl.value);
        },240,true,true,true);

        this.addSlider("lpfreq",\freq.asSpec,{
            arg sl;
            inSynth.set(\lpfreq,sl.value);
        },1200,true,true,true);

        this.addSlider("dbDown",\db.asSpec,{
            arg sl;
            dbDown = sl.value;
            filterGroup.set(\dbDown,dbDown);
        },-2,true,true,true);

        this.addSlider("fadeTime",ControlSpec(1,12),{
            arg sl;
            fadeTime = sl.value;
            filterGroup.set(\fadeTime,fadeTime);
        },4,true,true,true);

        this.addSlider("updateWait",ControlSpec(0.1,1),{
            arg sl;
            updateWaitTime = sl.value;
        },0.1,true,true,true);

        this.addSlider("waitMin",ControlSpec(1,50),{
            arg sl;
            waitMin = sl.value;
```

```
		},12,true,true,true);

		this.addSlider("waitMax",ControlSpec(1,50),{
			arg sl;
			waitMax = sl.value;
		},15,true,true,true);

		this.adjustWindowAndFront;
	}

	free {
		this.removeAllAssignments;
		outSynth.set(\gate,0);
		inSynth.free;
		mySpectrogram.free;
		AppClock.sched(0.04,{
			//inSynth.free;
			filterGroup.free;
			nil;
		});
		win.close;
	}

	inBus_ {
		arg inBus_;
		inBus = inBus_;
		inSynth.set(\inBus,inBus)
	}

	outBus_ {
		arg outBus_;
		outBus = outBus_;
		outSynth.set(\outBus,outBus);
	}

	pause {
		outSynth.set(\pauseGate,0);
		AppClock.sched(0.1,{
			inSynth.run(false);
			filterGroup.run(false);
			nil;
		});
	}

	run {
		inSynth.run;
		filterGroup.run;
		AppClock.sched(0.04,{
			outSynth.run;
			outSynth.set(\pauseGate,1);
			nil;
		});
	}

	/*  save {
	var saves;
	saves = Dictionary.new;

	^saves;
	}

	load {
	arg saves;
	}*/

```

```
    /* OPTIONAL CLASSES FOR INTERFACING WITH LEMUR

    lemurX
    lemurY
    lemurControlPad
    */

    lemurX {
        arg x;
    }

    lemurY {
        arg y;
    }

    lemurControlPad {
        arg cp;
    }
}
```

code/2/feedback amplification mod.sc

```
TptFdbk : ImprovModule {
    /* CLASS VARIABLES AND VARIABLES OF ImprovModule CLASS

    classvar <>server, >toLemur;
    var inBus, outBus, group, <cavity, <win, winBounds;

    */
    // these variables probably include the actual variables
    // of the module and also variables for each of the GUIs
    var <synth;

    /* METHODS THAT EACH MODULE MUST HAVE:

    initClass
    init {
    arg inBus_, outBus_, group_, cavity_;
    inBus = inBus_;
    outBus = outBus_;
    group = group_;
    cavity = cavity_;
    }

    free

    inBus_
    outBus_

    pause
    run

    save
    load

    */

    *initClass {
        StartUp.defer {
            SynthDef(\cm_feedback_4Chan,{
                arg mix = 1, inBus, outBus, gate, pauseGate, hpfreq = 400;
                var in, sig, amCoef, amount = 0.99;
                in = HPF.ar(In.ar(inBus,4), hpfreq)*5;

                amCoef = 2*amount/(1-amount);
```

```
                sig = MidEQ.ar(
                    LPF.ar((1+amCoef)*in/(1+(amCoef*in.abs)),[3800, 3900])*0.5,
                    120,
                    0.7,
                    8
                );
                //sig * -24.dbamp;
                sig = SelectX.ar(mix,[in,sig]);
                Out.ar(outBus,sig);
            }).writeDefFile;
        }
    }

    init {
        arg inBus_, outBus_, group_, cavity_;
        inBus = inBus_;
        outBus = outBus_;
        group = group_;
        cavity = cavity_;

        synth = Synth(\cm_feedback_4Chan,[\inBus,inBus,\outBus,outBus,\gate,1,\pauseGate,1],
            group);

        this.makeWindow;

        this.addSlider("hpfreq",\freq.asSpec,{
            arg sl;
            synth.set(\hpfreq,sl.value);
        },400,true,false,true);

        this.adjustWindowAndFront;
    }

    free {
        this.removeAllAssignments;
        synth.set(\gate,0);
        //win.close;
    }

    inBus_ {
        arg inBus_;
        inBus = inBus_;
        synth.set(\inBus,inBus)
    }

    outBus_ {
        arg outBus_;
        outBus = outBus_;
        synth.set(\outBus,outBus);
    }

    pause {
        synth.set(\pauseGate,0);
    }

    run {
        synth.run;
        synth.set(\pauseGate,1);
    }

/*  save {
        var saves;
        saves = Dictionary.new;

        ^saves;
```

```
    }

    load {
        arg saves;
    }*/

    /* OPTIONAL CLASSES FOR INTERFACING WITH LEMUR

    lemurX
    lemurY
    lemurControlPad
    */

    lemurX {
        arg x;
    }

    lemurY {
        arg y;
    }

    lemurControlPad {
        arg cp;
    }
}
```

E Code for Section 3.3.2

code/3/02_descriptors_extraction_func.scd

```
(
~extract_from_buf = {
    arg buf_path, slice_sec = 0.05, n_servers = 1, final_action;
    var stamp = Date.localtime.stamp;
    var prefix = PathName(buf_path).fileNameWithoutExtension;
    var wav_file_name = PathName(buf_path).fileName;
    var dsID_to_wavename_dict = Dictionary.new;
    var new_folder = PathName(buf_path).pathOnly+/+"%_%".format(stamp,prefix);
    var new_ds_folder = new_folder+/+"ds";
    var new_loc_ds_folder = new_folder+/+"loc_ds";
    var sR = SoundFile.use(buf_path,{arg sf;sf.sampleRate});
    var buf_dur = SoundFile.use(buf_path,{arg sf; sf.duration});
    var n_slices = (buf_dur / slice_sec).floor.asInteger;
    var full_array = Array.fill(n_slices,{
        arg i;
        var ds_id = "%-%".format(prefix,i.asInteger);
        var start_sec = i * slice_sec;
        dsID_to_wavename_dict.put(ds_id,wav_file_name);
        [ds_id, start_sec * sR, slice_sec * sR]; // id int, start frames, num frames
    });
    var n_derivs = 2; // it's really one though...
    var statsFlatComp = {
        arg featuresBuf,statsBuf,flatBuf,masterBuf,masterBufOffset,numDerivs=2,action;
        FluidBufStats.processBlocking(featuresBuf.server,featuresBuf,stats:statsBuf,numDerivs:numDerivs-1,action:{
            FluidBufFlatten.processBlocking(featuresBuf.server,statsBuf,flatBuf,action:{
                FluidBufCompose.processBlocking(featuresBuf.server,flatBuf,destination:masterBuf,destStartFrame:masterBufOffset,action:{
                    action.value;
                });
            });
        });
    };
    var analyze = {
```

```
        arg id, start_frame, num_frames, buf, featuresBuf,statsBuf, flatBuf,finalBuf, ds, cond;

        FluidBufSpectralShape.processBlocking(buf.server,buf,start_frame,num_frames,features:featuresBuf,action:{
            statsFlatComp.(featuresBuf,statsBuf,flatBuf,finalBuf,0,n_derivs,{
                FluidBufPitch.processBlocking(buf.server,buf,start_frame,num_frames,features:featuresBuf,action:{
                    statsFlatComp.(featuresBuf,statsBuf,flatBuf,finalBuf,98,n_derivs,{ // 98 = 7*7*2
                        FluidBufLoudness.processBlocking(buf.server,buf,start_frame,num_frames,features:featuresBuf,action:{
                            statsFlatComp.(featuresBuf,statsBuf,flatBuf,finalBuf,126,n_derivs,{ // 126 = 98 + 28; // 28 = (2*7*2)
                                FluidBufMFCC.processBlocking(buf.server,buf,start_frame,num_frames,features:featuresBuf,numCoeffs:40,action:{
                                    statsFlatComp.(featuresBuf,statsBuf,flatBuf,finalBuf,154,n_derivs,{// 154=126+28; // 28=(2*7*2) //
                                        ds.addPoint(id,finalBuf,{
                                            cond.unhang;
                                        });
                                    });
                                });
                            });
                        });
                    });
                });
            });
        });
    };
    var server_options = ServerOptions.new;
    var entries_per_sub = (full_array.size / n_servers).ceil.asInteger;
    // var sub_arrays = full_array.clump(entries_per_sub);
    var sub_arrays = List.new.dup(n_servers);
    var headers_expander = {
        arg header_list;
        var out_headers = List.new;
        n_derivs.do({
            arg deriv_num;
            ["mean","stddev","skewness","kurtosis","min","median","max"].do({
                arg stat;
                header_list.do({
                    arg desc;
                    out_headers.add("%-deriv%-%".format(desc,deriv_num,stat));
                });
            });
        });
        out_headers;
    };
    var finalHeaders = List.new;
    var buffer2mono = {
        arg buf, action;
        if(buf.numChannels > 1,{
            Routine{
                var new_buf = Buffer(buf.server);
                var total_chans = buf.numChannels;
                buf.server.sync;
                total_chans.do({
                    arg i;
                    FluidBufCompose.process(buf.server,buf,startChan:i,numChans:1,gain:total_chans.reciprocal,destination:new_buf,destGain:1);
                    buf.server.sync;
                });
                buf.free;
                action.(new_buf);
```

```
            }.play;
        },{
            action.(buf);
        });
    };

    full_array.do({
        arg pt, i;
        sub_arrays[i % n_servers].add(pt);
    });

    server_options.device_("Fireface UC Mac (24006457)");

    full_array.size.postln;
    sub_arrays.do({arg sa; sa.size.postln});

    finalHeaders.addAll(headers_expander.([
        "specCentroid",
        "specSpread",
        "specSkewness",
        "specKurtosis",
        "specRolloff",
        "specFlatness",
        "specCrest"
    ]));

    finalHeaders.addAll(headers_expander.(["pitch","pitchConf"]));

    finalHeaders.addAll(headers_expander.(["loudness","truePeak"]));

    finalHeaders.addAll(headers_expander.(Array.fill(40,{
        arg i;
        "mfcc%".format(i.asString.padLeft(2,"0"));
    })));

    finalHeaders = finalHeaders.collect({
        arg head, i;
        [i,head];
    });

    File.mkdir(new_folder);
    File.mkdir(new_ds_folder);
    File.mkdir(new_loc_ds_folder);
    ArrayToCSV(finalHeaders,new_folder+/+"%_%_headers.csv".format(stamp,prefix));
    dsID_to_wavename_dict.writeArchive(new_folder+/+"dsID_to_wavename_dict.sco");

    n_servers.do{
        arg server_i;
        var server = Server("Server-%-%".format(server_i,UniqueID.next).asSymbol,NetAddr("
            localhost",57121 + server_i),server_options);
        server.waitForBoot{
            Routine{
                var features_buf = Buffer(server);
                var stats_buf = Buffer(server);
                var flat_buf = Buffer(server);
                var final_buf = Buffer(server);
                var ds_ = FluidDataSet(server);
                var start_dur_ds = FluidDataSet(server);
                var start_dur_buf = Buffer.alloc(server,2);
                var whole_buf = Buffer.read(server,buf_path);
                var array = sub_arrays[server_i];
                var dspath = new_ds_folder+/+"%_%_server=%.json".format(stamp,prefix,
                    server_i);
                var start_dur_path = new_loc_ds_folder+/+"%_%_server=%_start_dur.json".
                    format(stamp,prefix,server_i);
```

```
                    server.sync;

                    "whole buf loaded".postln;

                    buffer2mono.(whole_buf,{
                        arg buf;
                        "mono buf made".postln;
                        array.do({
                            arg pt, i;
                            var cond = Condition.new;
                            var pt_label = pt[0];
                            start_dur_buf.setn(0,[pt[1],pt[2]]);
                            server.sync;
                            start_dur_ds.addPoint(pt_label,start_dur_buf,{
                                analyze.(pt_label,pt[1],pt[2],buf,features_buf,stats_buf,
                                    flat_buf,final_buf,ds_,cond);
                            });
                            cond.hang;
                            "id: % / %\t\t% / %".format(pt_label,full_array.size,i,array.size).
                                postln;
                        });

                        ds_.write(dspath,{
                            start_dur_ds.write(start_dur_path,{
                                // ds_.print;
                                // finalHeaders.postln;
                                // finalHeaders.size.postln;
                                /*"started:  %".format(stamp);
                                "finished: %".format(Date.localtime.stamp);
                                */
                                Routine{
                                    server.quit;
                                    10.wait;
                                    if(server_i == 0,{
                                        final_action.value;
                                    });
                                }.play;
                            });
                        });
                    });
                }.play;
            };
        };
};
);
```

code/3/03_descriptors_extraction.scd

```
(
~files = [
/*  "/Volumes/Ted's 10TB My Book (June 2020)/Research/machine learning/timbral_space_mapping
    /benjolin/outputs/210319_125601/chunk_1_MONO.wav",
    "/Volumes/Ted's 10TB My Book (June 2020)/Research/machine learning/timbral_space_mapping
        /benjolin/outputs/210319_125601/chunk_2_MONO.wav",*/
    "/Volumes/Ted's 10TB My Book (June 2020)/Research/machine learning/timbral_space_mapping
        /benjolin/outputs/210319_125601/chunk_3_MONO.wav",
    "/Volumes/Ted's 10TB My Book (June 2020)/Research/machine learning/timbral_space_mapping
        /benjolin/outputs/210319_125601/chunk_4_MONO.wav"
];

~f_r = {
    arg array, idx = 0;
    if(idx < array.size,{
        ~extract_from_buf.(array[idx],1,3,{
```

```
            ~f_r.(array,idx+1);
        });
    });
};
)

(
~f_r.(~files);
)
```

code/3/04_compile_ds.scd

```
(
s.options.device_("Fireface UC Mac (24006457)");
s.waitForBoot{
    var folders = [
        "/Volumes/Ted's 10TB My Book (June 2020)/PROJECT FILES/SWITCH/media/wavs/
            _analyses_210412_02_sliceSize=0.1/210412_151245_210408_221536_creatures_tj/",
        "/Volumes/Ted's 10TB My Book (June 2020)/PROJECT FILES/SWITCH/media/wavs/
            _analyses_210412_02_sliceSize=0.1/210412_151644_210408_221536_shoe_squeak_tj/"
    ];

    var out_folder = "/Volumes/Ted's 10TB My Book (June 2020)/PROJECT FILES/SWITCH/media/
        wavs/_analyses_210412_02_sliceSize=0.1/_selections_for_creatures_tj/";

    Routine{
        var master_ds = FluidDataSet(s);
        var master_loc_ds = FluidDataSet(s);
        var temp_ds = FluidDataSet(s);
        var master_ds_id_to_wavname = Dictionary.new;

        s.sync;

        folders.do({
            arg folder, i;
            var dsid2wav = Object.readArchive(folder+/+"dsID_to_wavename_dict.sco");

            folder.postln;

            dsid2wav.keysValuesDo({
                arg key, val;
                master_ds_id_to_wavname.put(key,val);
            });

            PathName(folder+/+"ds/").filesDo({
                arg file, j;
                var cond = Condition.new;

                "---%".format(file.fullPath).postln;

                if((i == 0) && (j == 0),{
                    master_ds.read(file.fullPath,{cond.unhang});
                },{
                    temp_ds.read(file.fullPath,{
                        master_ds.merge(temp_ds,0,{
                            cond.unhang;
                        });
                    });
                });
                cond.hang;
            });

            PathName(folder+/+"loc_ds/").filesDo({
                arg file, j;
                var cond = Condition.new;
```

```
                "---%".format(file.fullPath).postln;

                if((i == 0) && (j == 0),{
                    master_loc_ds.read(file.fullPath,{cond.unhang});
                },{
                    temp_ds.read(file.fullPath,{
                        master_loc_ds.merge(temp_ds,0,{
                            cond.unhang;
                        });
                    });
                });
                cond.hang;
            });
        });

        File.mkdir(out_folder);

        master_ds.write(out_folder+/+"ds.json",{
            master_loc_ds.write(out_folder+/+"loc_ds.json",{
                master_ds_id_to_wavname.writeArchive(out_folder+/+"ds_id_to_wavname.sco");
                ArrayToCSV(folders,out_folder+/+"all_input_folders.csv");
                "done".postln;
            });
        });
    }.play;
};
)
```

code/3/05_dim_reduction.scd

```
(
s.options.device_("Fireface UC Mac (24006457)");
s.waitForBoot{
    Routine{
        var compiled_folder = "/Volumes/Ted's 10TB My Book (June 2020)/PROJECT FILES/SWITCH/
            media/wavs/_analyses_210412_02_sliceSize=0.1/_instruments_only/";

        var loudness_thresh = -60;
        //              SpecShape   pitch, pitchCon,    mfcc 1-9
        //var select_cols = (0..6) ++ [98,99] ++ (155..163);
        var select_cols = (0..713);

        var ds = FluidDataSet(s);
        var scaler = FluidStandardize(s);
        var dsq = FluidDataSetQuery(s);
        var pcaDims = 35;
        var pca = FluidPCA(s,pcaDims);
        var umapDims = 1;
        var umapNeighbors = 30;
        var umapMinDist = 0.5;
        var umap = FluidUMAP(s,umapDims,umapNeighbors,umapMinDist);

        s.sync;

        ds.read(compiled_folder+/+"ds.json",{
            ds.cols({
                arg n_cols;
                dsq.addRange(0,n_cols,{
                    dsq.filter(126,">",loudness_thresh,{
                        dsq.transform(ds,ds,{
                            ds.size({
                                arg size;
                                "size after loudness filter: %".format(size).postln;
                                dsq.clear({
```

```
                                    Routine{
                                        select_cols.do({
                                            arg col;
                                            dsq.addColumn(col);
                                            s.sync;
                                        });

                                        dsq.transform(ds,ds,{
                                            scaler.fitTransform(ds,ds,{
                                                pca.fitTransform(ds,ds,{
                                                    "pca done".postln;
                                                    umap.fitTransform(ds,ds,{
                                                        "umap done".postln;
                                                        ds.write(compiled_folder+/+"%_pca=%-%-%.json".format(Date.localtime.stamp,umapDims,umapNeighbors,umapMinDist));
                                                    });
                                                });
                                            });
                                        });
                                    }.play;
                                });
                            });
                        });
                    });
                });
            });
        });
    }.play;
};
)
```

code/3/13_sort_umap1.scd

```
(
~restart_oscdef.value;
s.waitForBoot{
    Routine{
        var loc_ds_path = "/Volumes/Ted's 10TB My Book (June 2020)/PROJECT FILES/SWITCH/media/wavs/_analyses_210412_02_sliceSize=0.1/_instruments_only/loc_ds.json";
        var ds_path = "/Volumes/Ted's 10TB My Book (June 2020)/PROJECT FILES/SWITCH/media/wavs/_analyses_210412_02_sliceSize=0.1/_instruments_only/210415_232342_umap=1-30-0.5.json";
        var ds_id_to_wavname_path = "/Volumes/Ted's 10TB My Book (June 2020)/PROJECT FILES/SWITCH/media/wavs/_analyses_210412_02_sliceSize=0.1/_instruments_only/ds_id_to_wavname.sco";

        var ds = FluidDataSet(s);
        var loc_ds = FluidDataSet(s);

        s.sync;

        loc_ds.read(loc_ds_path,{
            loc_ds.dump({
                arg loc_dict;
                ds.read(ds_path,{
                    ds.dump({
                        arg dict;
                        var pts = Array.newClear(dict.at("data").size);
                        var ds_d_to_wavname = Object.readArchive(ds_id_to_wavname_path);

                        dict.at("data").keysValuesDo({
                            arg key, val, i;
                            pts[i] = [val[0],key];
```

```
                        });

                        pts.sort({
                            arg a, b;
                            a[0] < b[0];
                        });

                        pts.postln;

                        Routine{
                            var start_pct = 0.0, end_pct = 1;
                            var start = (pts.size * start_pct).asInteger;
                            var end = (pts.size * end_pct).asInteger - 1;
                            pts[start..end].do({
                                arg array, i;
                                var loc = (i % 4) + 2;
                                var id = array[1];
                                ~playID.(id,ds_d_to_wavname,loc_dict,loc:loc);
                                0.02.wait;
                            });
                        }.play;
                    });
                })
            });
        });
    }.play;
}
)
```

code/3/14_pca.scd

```
(
s.options.device_("Fireface UC Mac (24006457)");
s.waitForBoot{
    Routine{
        var compiled_folder = "/Volumes/Ted's 10TB My Book (June 2020)/PROJECT FILES/SWITCH/
            media/wavs/_analyses_210412_02_sliceSize=0.1/_instruments_only/";

        var loudness_thresh = -60;

        var ds = FluidDataSet(s);
        var scaler = FluidStandardize(s);
        var dsq = FluidDataSetQuery(s);
        var pcaDims = 11;
        var pca = FluidPCA(s,pcaDims);

        s.sync;

        ds.read(compiled_folder+/+"ds.json",{
            ds.cols({
                arg n_cols;
                dsq.addRange(0,n_cols,{
                    dsq.filter(126,">",loudness_thresh,{
                        dsq.transform(ds,ds,{
                            ds.size({
                                arg size;
                                "size after loudness filter: %".format(size).postln;
                                scaler.fitTransform(ds,ds,{
                                    pca.fitTransform(ds,ds,{
                                        var pts = List.new, labels = List.new, stamp = Date.
                                            localtime.stamp;
                                        "pca done".postln;
                                        ds.write(compiled_folder+/+"%_pca=%_ds.json".format(
                                            stamp,pcaDims));
                                        pca.write(compiled_folder+/+"%_pca=%_pca.json".
```

```
                                            format(stamp,pcaDims));
                                        ds.dump({
                                            arg dict;
                                            dict.at("data").keysValuesDo({
                                                arg key, val;
                                                labels.add(key);
                                                pts.add(val);
                                            });
                                            ArrayToCSV(labels,compiled_folder+/+"%_pca=%_labels.csv".format(stamp,pcaDims));
                                            ArrayToCSV(pts,compiled_folder+/+"%_pca=%_pts.csv".format(stamp,pcaDims));
                                        });

                                        /*pca.dump({
                                            arg dict;
                                            dict.postln;
                                        });*/
                                    });
                                });
                            });
                        });

                    });
                });
            });
        });
    }.play;
};
)
```

code/3/15_tsp_to_sound.scd

```
(
~restart_oscdef.value;
s.waitForBoot{
    Routine{
        var pythonOutPath = "/Volumes/Ted's 10TB My Book (June 2020)/PROJECT FILES/SWITCH/media/wavs/_analyses_210412_02_sliceSize=0.1/_instruments_only/210416_004952_pca=11_pts_tspOutput.csv";
        var label_order_path = "/Volumes/Ted's 10TB My Book (June 2020)/PROJECT FILES/SWITCH/media/wavs/_analyses_210412_02_sliceSize=0.1/_instruments_only/210416_004952_pca=11_labels.csv";
        var ds_id_to_wavname_path = "/Volumes/Ted's 10TB My Book (June 2020)/PROJECT FILES/SWITCH/media/wavs/_analyses_210412_02_sliceSize=0.1/_instruments_only/ds_id_to_wavname.sco";
        var loc_ds_path = "/Volumes/Ted's 10TB My Book (June 2020)/PROJECT FILES/SWITCH/media/wavs/_analyses_210412_02_sliceSize=0.1/_instruments_only/loc_ds.json";

        var loc_dict;
        var loc_ds = FluidDataSet(s);
        var ds_id_to_wavname = Object.readArchive(ds_id_to_wavname_path);

        var pythonOutData = CSVFileReader.readInterpret(pythonOutPath,true);
        var label_order = CSVFileReader.read(label_order_path,true);
        var orderedFrames = pythonOutData[0];
        var distances = pythonOutData[1];
        var tsp_label_order = Array.newClear(orderedFrames.size);

        s.sync;

        loc_ds.read(loc_ds_path,{
            loc_ds.dump({
                arg loc_dict;

```

```
                tsp_label_order = orderedFrames.collect({
                    arg idx;
                    label_order[idx][0];
                });

                Routine{
                    // var waitTime = (30.reciprocal / 4);
                    tsp_label_order.do({
                        arg id, i;
                        var loc = (i % 4) + 2;
                        ~playID.(id,ds_id_to_wavname,loc_dict,0.1,loc:loc);

                        0.02.wait;

                    });
                }.play;
            });
        });
    }.play
}
)
```