Circle: High-Dimensional Latent Space Traversal in Real Time on a Two-Dimensional Controller

Matthew Creighton
Goldsmiths, University of London
mcrei001@gold.ac.uk*
matthew.k.creighton@gmail.com

Abstract

This paper describes a novel musical controller for high-dimensional latent space traversal in real time. Control is precise, repeatable and explicit, and the structure of the control space affords mediation between different latent spaces and models.

We introduce the Circle Gesture Language: a family of two-dimensional periodic curves, with a parametrization that associates each curve with a point in a high-dimensional space. By continuously tracing such a curve, a user can refer to points in the high-D space using a 2-D controller.

We present Circle∼, a MaxMSP package which recognizes these gestures in real time and derives the high-D coordinate vector. We include a mouse-based controller, a testbed, and a demonstration controlling a well-known synthesis tool in 26 dimensions.

We evaluate these contributions from a technical point of view and reflect on their artistic potential.

1 INTRODUCTION

From VAE-based neural synthesis tools like RAVE (Caillon and Esling, 2021) and its various refinements (Caspe et al., 2025), to methods based on differential DSP (Engel et al., 2020) or concatenative synthesis (Thibault et al., 2025), to learnable wavetables (Shan et al., 2022), modern AI music tools are replete with large latent spaces and high-dimensional feature vectors - whether those high-D representations are normally accessible to performers or not. In such a landscape the methods of traversal within latent spaces are as important as the processes of data collection and training that bring them into being.

In this paper we focus on tools for *direct live traversal* of latent spaces, involving no music agents or trajectory generating tools, where moment-to-moment agency is entirely in the hands of the performer. We argue for a more transferrable and generic toolset for high-D direct traversal in a range of settings, and discuss the development of one such tool.

Dimensionality reduction is the mapping of a large space onto a smaller space in such a way that desired features within the larger space are preserved. Peachey et al. (2025), Zheng et al. (2025) and Martelloni and Kiefer (2025) are among many to reduce all the way to two or three dimensions, in order that the reduced space can be literally mapped into physical space or the state space of a controller. These *manifold interfaces* (Choi, 1998) provide an immediate and intuitive interface but are limited to a handful of dimensions.

How can we break out of three dimensions? *Timbre transfer* is the process of analyzing an incoming sound, extracting parameters from it, and using those parameters to control a synthesis process, as in Caillon and Esling (2021) or Caspe et al. (2025). Dimensionality reduction is still required but the dimension of the reduced space can be higher, and existing instruments and sound sources can be used as controllers. The user does not directly perceive the structure of the latent space, however, instead interacting with the system as a whole.

How can we design a more transparent and navigable tool, in which the latent space is sensed as an object? Tahiroğlu et al. (2021) and Privato et al. (2024) discuss approaches which might be referred to as *tactile direct traversal*, building electromechanical systems with many degrees of freedom and deriving control parameters from the measured physical state of the controller. These interfaces are navigable and allow for structured exploration of the sound space, but tend to be tightly coupled to a particular application (e.g. with respect to the number of tracked parameters)

How can we build a more generic and higher dimensional tool? To gain some purchase on this question we consider the transforms involved and put together a *space proposition* (Choi, 1998), working from Arfib et al. (2002) and Mulder (1998). See fig. 1 - *Sensor Space* is the state space of the controller. *Feature Space* is the intermediate mapping space - one can think of this as the space of all semantically meaningful "instantaneous control actions" - and *Phase Space* is the parameter space of the the sounding element (e.g. a synthesis tool).

The approaches discussed so far either identify the sensor space directly with the feature space (e.g. manifold interfaces), or have a low-dimensional feature space. The lower dimensionality of the feature space can be mapped to the high-D phase space using dimensionality reduction, but this attaches us to a low-D manifold in the phase space - not all points are reachable and not all directions are explorable.

How can we create a higher dimensionality in the feature space? Tools like Wekinator (Fiebrink and Cook, 2010) and techniques like play along mapping (Fiebrink et al., 2009) allow for complex many-to-many mappings - meaning sensor or controller input can itself be high-dimensional and complex. This can be mapped into a larger region in the phase space. Mappings have to be relearned for each new environment, however.

*invalid from mid-2027 onwards

Figure 1: Transformation chain from physical interaction to sound synthesis.

Can we create a high-D feature space without special high-D sensor hardware or retraining in each new environment? And can we meet the other requirements of transparent navigability and genericness?

With these questions in mind we designed the Circle Gesture Language (CGL) - a set of 2D periodic curves which can be drawn with a tablet or mouse, with a transform taking each curve to a unique high dimensional vector. While the dimension of the sensor space is low, it can map to all points in a large feature space because curves drawn over time have many more degrees of freedom than individual points in a space. Since we design the gesture language explicitly rather than learning from input, there is no training step and skills with the system are transferable between latent spaces.

We refer to the 2D periodic figures as *gestures* or *curves* and the high-dimensional vectors as *feature vectors* or simply *features*.

CGL is:

- **Real-Time**
- **Deterministic and synactically explicit** - the same gesture reliably gives the same feature vector, and skilled users can move in a specific direction in the feature space at will.
- **Continuous** - gestures are cyclic and can be drawn indefinitely. Continuous paths in the feature space are traced by continuous transforms of the gesture.
- **Discontinuous-at-will** - there is no obligation to make only continuous changes to the gesture. Jumps of arbitrary size and direction can be made within the feature space, creating simultaneous changes to a large number of parameters.
- **High-dimensional** - there is no theoretical limit to the number of feature space dimensions. We include a musical demonstration with 26 dimensions.
- **Rooted and (long-sense) stateless** - no permanent state is retained. If the user stops moving, the system returns (with time) to its "root state" with all parameters 0.
- **Segmented** as will be discussed, the feature space naturally divides itself into smaller sets of parameters, helping to avoid overwhelm and opening the possibility for navigating though multiple latent spaces simultaneously.
- **Human-drawable** - we use an acceleration measure as a stand-in for "easily drawable", and show that the CGL behaves well in this respect. We also demonstrate the drawability of a wide range of shapes in practice through a video demonstration.
- **Preprocess-free** - no learning stage. The input is 2D and easy to connect to a wide range of controllers.

Drawbacks to this approach include:

- **Not fully independent** - There are limits on the values the feature space vector can take - in particular limits to how many parameters can be nonzero at one time. (although see section 4.2.1'Multiphonics, Transients and Smoothing Time')
- **Nonzero latency** - The gesture language centers around a 2-D frequency analysis of the incoming gesture. This means it takes at least one cycle of the gesture for the system to "lock on". This latency is variable - faster gestures are more responsive, and slower gestures less so.

2 FORMULATING THE GESTURE LANGUAGE

We consider two spaces - the two-dimensional *sensor space* or *curve space*, and the higher dimensional *feature space*.

For our purposes, a gesture $z(t)$ is a continuous, periodic function from t (time) to the curve space - that is, a closed 2-D shape that the user can trace repeatedly. We treat the two dimensions of the curve space as a single complex dimension.

Each gesture is associated with a single point in the feature space, with coordinate vector $\boldsymbol{Z}$. We want the transform from curve space to feature space to be continuous - that is, small changes to $z(t)$ should not induce discontinuities in $\boldsymbol{Z}$ or vice versa.

What family of gestures should we consider? The curves should be "easy to draw" - smooth, not sharp and jerky. That is, there shouldn't be sudden spikes in acceleration.

we can formalize this by considering the periodic $z(t)$ as a sum of sinusoids:

$$z(t) = \sum_k A_k e^{j(\omega_k t)} \tag{1}$$

and considering the second derivative (the acceleration):

$$z''(t) = -\sum_k A_k \omega_k^2 e^{j(\omega_k t)} \tag{2}$$

which is bound from above by:

$$\max |z''(t)| \leq \sum_k A_k w_k^2 \tag{3}$$

For a given amplitude scale (set by the size of the controller) and frequency band (set by the agility of the hand), we can minimize jerkiness (peak acceleration) by minimizing k - that is, by enforcing *frequency sparsity*. We could say that sparser curves are easier to draw, independently of the body or process which draws them.

The definition of 2-Ellipse curves (discussed below) and the resulting CGL vector $\boldsymbol{Z}$ follow from a strict focus on sparsity and periodicity. We choose these constraints because they make drawability and continuity easier to guarantee. By altering them, other curve families and thus other gesture languages might be derived.

2.1 Ellipses

Consider a very sparse family of curves - the curves with information at only a single (magnitude) angular frequency $|w|$. In 2 dimensions the negative (anticlockwise) frequency information is independent of the positive (clockwise) information, so this is the sum of two sinusoids:

$$z(t) = a_\ell e^{j(\omega t+\phi_\ell)} + a_r e^{-j(\omega t+\phi_r)} \tag{4}$$

$z(t)$ traces out an ellipse - hence "1-Ellipse curve".

By using a sum-and-difference substitution

$$\tau = \frac{\phi_\ell - \phi_r}{2}, \quad \psi = \frac{\phi_\ell + \phi_r}{2} \tag{5}$$

we separate the phase difference τ from the common starting phase ψ:

$$z(t) = a_\ell e^{j(\omega t+\psi+\tau)} + a_r e^{-j(\omega t+\psi-\tau)} \tag{6}$$

Observe that changes in ψ are equivalent to time offsets, since

$$\begin{aligned} z\Big(t - \frac{\psi}{\omega}\Big) &= a_\ell e^{j(\omega t+\tau)} + a_r e^{j(-\omega t-\tau)} \\ &= z(t)_{\psi=0} \end{aligned} \tag{7}$$

for all ψ. Thus if we are only interested in shape we can discard ψ . (not so for sums of ellipses - see section 2.2 '2-Ellipse curves').

we can describe the amplitudes a_ℓ, a_r using a scale-invariant shape parameter S and a pure-scale parameter A:

$$A = \frac{(a_\ell + a_r)}{2}, \quad S = \frac{a_\ell - a_r}{a_\ell + a_r} \tag{8}$$

These equations invert to

$$a_\ell = A(1+S), \quad a_r = A(1-S) \tag{9}$$

giving a 4-dimensional ellipse representation as follows:

$$\begin{aligned} &z_4(\omega, A, S, \tau)(t) = \\ &\quad A(1+S)e^{j(\omega t+\tau)} + A(1-S)e^{j(-\omega t-\tau)} \end{aligned} \tag{10}$$

This curve approaches an anticlockwise circle as S approaches 1, a straight line segment as S approaches zero, and a clockwise circle as S approaches -1. For $|S| < 1$, τ gives the angle of the major axis of the ellipse with the x axis. A gives the amplitude and ω the (angular) frequency. $180°$ flips in τ are ambiguous with changes in ψ, which we prefer not to track - to resolve this we limit τ into $-\frac{\pi}{2} \le \tau < \frac{\pi}{2}$. Figure 2a shows the effect of the S and τ parameters on the curve shape.

By tracing curves in this curve family, a user can refer to points in 4-D feature space from the 2-D curve space. We can think of $z_4(\cdot)$ as a 2D-to-4D *gesture language*.

2.2 2-Ellipse curves

To describe a more diverse set of curves while maintaining sparsity, we can sum two ellipses at different frequencies.

$$\begin{aligned} z_8(t) =& z_4(\omega_1, A_1, S_1, \tau_1)(t) \\ &+ z_4(\omega_2, A_2, S_2, \tau_2)(t) \end{aligned} \tag{11}$$

By relativizing frequency:

$$\omega_0 = \omega_1, \quad \omega_r = \frac{\omega_2}{\omega_1} \tag{12}$$

and amplitude:

$$A_0 = A_1 + A_2 \tag{13}$$

$$A_{bal} = \frac{A_1}{A_0} \tag{14}$$

we arrive at:

$$\begin{aligned} z_8(t) = z_4\Big(\omega_0,\ A_0{\cdot}A_{bal},\ S_1,\ \tau_1\Big)(t) \\ + z_4\Big(\omega_0,\ A_0{\cdot}(1{-}A_{bal}),\ S_2,\ \tau_2\Big)(t) \end{aligned} \tag{15}$$

Equation (7) showed that the shape of $z_4(\cdot)$ was independent of ψ since changes in ψ are equivalent to time offsets. Equation (15), however, has *two* instances of $z_4(\cdot)$ on the right hand side: their relative starting phase affects the shape even though the individual ψ-values do not. This is illustrated in fig. 2c. To account for this without disturbing our ψ-free representation we add a parameter:

$$\phi = \psi_1 - \psi_2 \tag{16}$$

and represent it in the model as a time offset relative to the ψ-free $z_4(\cdot)$

$$\begin{aligned} z_9(t) = z_4\Big(\omega_0,\ A_0 A_{bal},\ S_1,\ \tau_1\Big)\Big(t - \phi/\omega_0\Big) \\ + z_4\Big(\omega_0\,\omega_r,\ A_0\,(1{-}A_{bal}),\ S_2,\ \tau_2\Big)\Big(\,t\,\Big) \end{aligned} \tag{17}$$

$z_9(\cdot)$, which could be called a 2D-to-9D gesture language, describes a very diverse set of curves (fig. 2b). In fact every curve considered in this paper can be described with some combination of the 9 parameters $\omega_r, \omega_0, A_0, A_{bal}, S_1, S_2, \tau_1, \tau_2$, and ϕ .

However - the ω_r parameter behaves differently to the others with respect to continuity. This causes some practical difficulties, but also creates the unique properties of our high-D segmented feature space.

2.3 Continuity and ω_r

It is a design principle of the Circle Gesture Language that all the curves described should be periodic. This means a user can trace them continuously and thus hold a particular state for an indefinite period of time.

For two sinusoids at different frequencies, their combined period is the least-common-multiple of the individual periods. The $lcm(\cdot)$ is unbounded over any interval - consider that

$$lcm(2, 3) = 6 \tag{18}$$

while

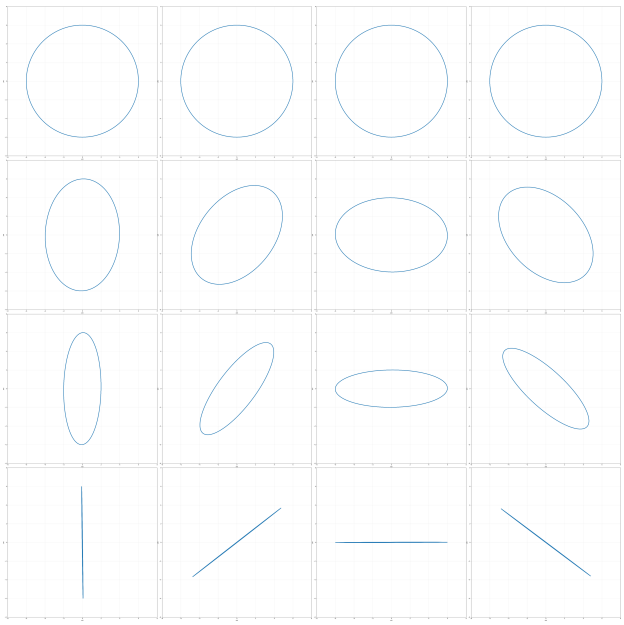
$$lcm(\frac{2000001}{2000000}, 3) = 2000001 \tag{19}$$

This means that any continuous path though values of ω_r refers to curves with arbitrarily long period, which are hard to notate, recognize and draw.

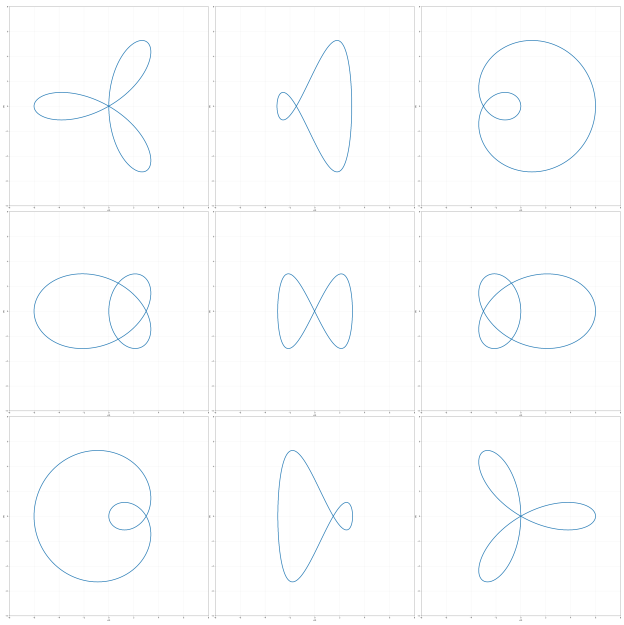
Our solution to this is to view ω_r as a rational number:

$$\omega_r = N/M \tag{20}$$

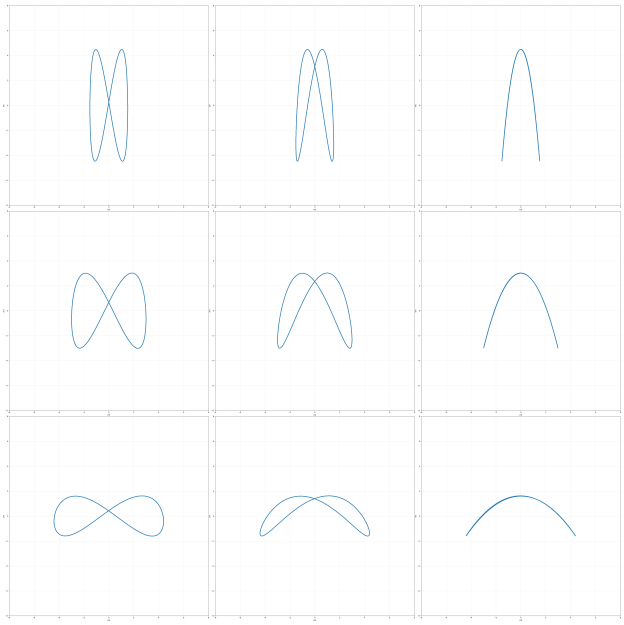
and simply disallow all M larger than some value, ensuring that the LCM remains bounded. In the current implementation $M_{max} = 2$, with some internal scaffolding in place for $M_{max} \in \{3, 4\}$ should it be required. We also - for practical reasons and to simplify the below derivation - choose some N_{max} - currently 5.



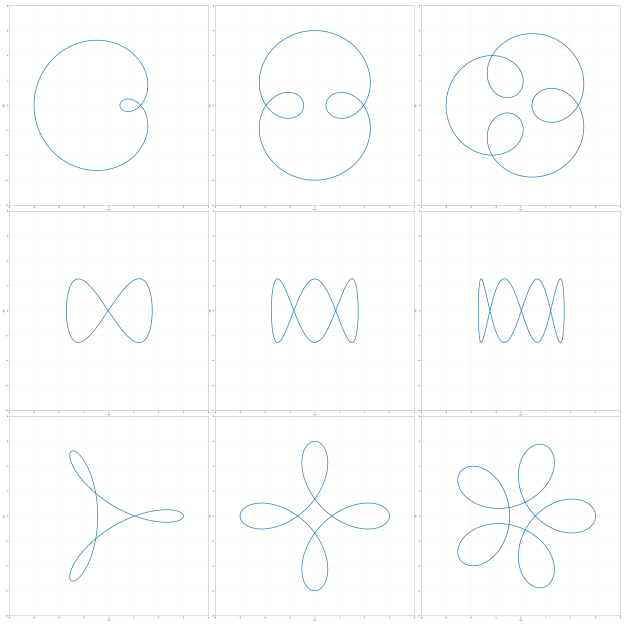
(a) Effect of S and τ values on the curve shape in the $z_4(\cdot)$ single ellipse model. S increases bottom-to-top from 0 to 1. τ increases left-to-right from $-\frac{\pi}{2}$ to $\frac{\pi}{4}$. A and ω affect scale and frequency only - so this figure depicts all of the "shape space" of $z_4(\cdot)$.



(b) The effect of parameters S_1 and S_2 on the curve shape in the $z_9(\cdot)$ 2-Ellipse model with $\omega_r = 2$. S_1 increases bottom-to-top from -1 to 0 to 1. S_2 increases left-to-right from -1 to 0 to 1. Other parameters are fixed at: $A_{bal} = 0.5$, $\tau_1 = 0$, $\tau_2 = \frac{\pi}{2}$. $\phi = \frac{\pi}{2}$



(c) The effect of parameters A_{bal} and ϕ on the curve shape in the $z_9(\cdot)$ 2-Ellipse model. A_{bal} increases bottom-to-top from 0.25 to 0.5 to 0.75. ϕ increases left-to-right from $-\frac{\pi}{2}$ to $-\frac{\pi}{4}$ to 0. Other parameters are fixed at: $\omega_r = 2, S_1 = S_2 = 0, \tau_1 = 0, \tau_2 = \frac{\pi}{2}$.



(d) Effect of ω_r and S_1, S_2 values on the curve shape in the $z_9(\cdot)$ two ellipse model. ω_r increases left-to-right from 2 to 4. Top row: S_1 and S_2 have the same sign. Middle row: $S_1 = S_2 = 0$. Bottom row: S_1 and S_2 have opposite sign. Other parameters are fixed at: $A_{bal} = 0.4$, $\tau_1 = 0$, $\tau_2 = \frac{\pi}{2}$. $\phi = \frac{\pi}{2}$

Figure 2: A selection of curves from the Circle Gesture Language.

This has far reaching consequences. The 2-Ellipse space is now *segmented* - split into a series of smaller spaces, one for each allowed ω_r value. These spaces still have 8 degrees of freedom (being the other 8 parameters of $z_9(\cdot)$) , but they are mostly disconnected from one another. A continuous[1] path from one ω_r-subspace to another has to "fade out" one ellipse completely (by reaching $A_{bal} = 0$ or $A_{bal} = 1$) before "fading in" another at a different frequency. Put another way - all continuous paths between differing ω_r-subspaces have to pass through 1-Ellipse space.

At this point a flaw in the $z_9(\cdot)$ representation is revealed. When $A_{bal} = 0$ or $A_{bal} = 1$, one of the two underlying ellipses has zero amplitude. So the shape and frequency parameters $\{S, \tau, \omega\}$ for that ellipse are *underdetermined* - they can take any value without changing the corresponding 2D curve. Approaching 1-Ellipse space from one of the ω_r-subspaces, the shape-frequency parameters are fixed to definite values. Leaving 1-Ellipse space into a different ω_r-subspace, the shape and frequency parameters are fixed to a *different* set of values. This causes a discontinuous jump in feature space - even though the changes to the 2D curve were continuous. The coordinates in different ω_r-subspaces disagree at their shared boundary in 1-Ellipse space.

To resolve this thorny topological issue, let us first observe that for every curve of interest we have some parametrization which works well:

- curves within each ω_r-subspace are well described in 9-D by $z_9(\cdot)$
- curves with just one frequency component are well described in 4-D by $z_4(\cdot)$

But:

- curves with just one frequency component are **not** well described in 9-D by $z_9(\cdot)$ due to underdetermination

Therefore we associate each ω_r-subspace with its own separate coordinate vector $\boldsymbol{Z}_{\omega r}$ - i.e. $\boldsymbol{Z}_2$, $\boldsymbol{Z}_{3/2}$ etc. The components of these vectors are equal to the usual $z_9(\cdot)$ coordinates within their respective subspaces, and zero otherwise. Similarly, we associate a 4-dimensional $\boldsymbol{Z}_1$ vector with the 1-Ellipse space, who's values are the usual $z_4(\cdot)$ coordinates in 1-Ellipse space and 0 otherwise. Concatenate them into a very-high-dimensional vector $\boldsymbol{Z}$:

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{Z}_1 \\ \boldsymbol{Z}_{3/2} \\ \boldsymbol{Z}_2 \\ \boldsymbol{Z}_{5/2} \\ \vdots \\ \boldsymbol{Z}_5 \end{bmatrix}$$

We have solved the problem of 1-Ellipse disagreement by brute force: for single ellipse curves, $\boldsymbol{Z_1}$ is the only non-zero component of $\boldsymbol{Z}$ by definition. But many discontinuities remain as we move between subspaces and the coordinates of one component or another suddenly jump to zero.

To resolve this we can ease the coordinates in each $\boldsymbol{Z_k}$, $k > 1$ towards zero near to 1-ellipse space, and ease the $\boldsymbol{Z_1}$ up from zero in those places by projecting to the nearest point in 1-Ellipse space, then scaling that value based on the distance from 1-Ellipse space. One could choose easing functions which sum to 1 at all points so that energy is preserved in some sense.

Thus we resolve the rational-ω_r and underdetermination issues by reparameterizing the same space with a very large number of dimensions, along with a constraint around which dimensions can be simultaneously non-zero.

The software implementation, rather than defining an explicit easing function based on distance, hunts for each ω_r feature in the incoming data and keeps track of the share of the (abstractly defined) "feature energy" it takes up - see section 3.2.4 'Instantaneous Feature Extraction'. The effect is similar - however, this approach allows virtuosic users to draw 3-Ellipse or 4-Ellipse curves to activate multiple features at once, partially transcending the restriction on simultaneous nonzero parameters.

2.4 Cartesian shape coordinates

The last hurdle before we arrive at a full definition of the CGL is to make sure that "easing the coordinates down towards zero" is a meaningful operation for the space within which we work. τ_1 and τ_2 provide problems in this respect, since they represent angles - it's not clear where the "zero point" should be. τ is also entangled with S, being undefined when $|S| = 1$, and unstable near that point.

we can address these problems with a "cartesian" representation which refers only to magnitudes and not angles. First define:

$$Q_{hv} = \cos(2\tau), \quad Q_{ud} = \sin(2\tau) \tag{21}$$

(recall that τ is limited to $-\frac{\pi}{2} \leq \tau < \frac{\pi}{2}$, hence the 2τ in eq. (21) to ensure continuity and orthogonality)

τ values are most well-defined when S is small, and should scale down for large S. Define:

$$Q_{sum} = |Q_{hv}| + |Q_{ud}| \tag{22}$$

$$R = 1 - |S| \tag{23}$$

And use the following 6-dimensional cartesian representation for τ, S:

$$\begin{aligned} S^+ &= \max(S, 0) \\ S^- &= \max(-S, 0) \\ T_h &= \tfrac{R}{Q_{sum}} \cdot \max(Q_{hv}, 0) \\ T_v &= \tfrac{R}{Q_{sum}} \cdot \max(-Q_{hv}, 0) \\ T_u &= \tfrac{R}{Q_{sum}} \cdot \max(Q_{ud}, 0) \\ T_d &= \tfrac{R}{Q_{sum}} \cdot \max(-Q_{ud}, 0) \end{aligned} \tag{24}$$

These 6 values always sum to 1, and like τ, S they are scale invariant. They all fit in the range 0-1, which makes the scale-to-zero operation described in section 2.3 'Continuity and ω_r' work smoothly. S^+ and S^- represent the "anticlockwiseness" and "clockwiseness" respectively, and T_h, T_v, T_u, T_d represent "horizontalness", "verticalness", closeness to the "upward diagonal" with positive gradient, and closeness to the "downward diagonal" with negative gradient, respectively.

this brings the dimensionality for a single ω_r-subspace to 17: $\omega_r, \omega_0, A_0, A_{bal}, \phi$, plus 6 shape parameters for the high-frequency ellipse, and 6 for the low-frequency ellipse.

[1] from the point of view of curve space/sensor space

These can be substituted into $\boldsymbol{Z}$ in the same manner as the $z_9(\cdot)$ representation.

we refer to this representation - a number of ω_r subspaces set by the chosen N, M, each parametrized in the 17-dimensional cartesian representation, plus a similar representation for the 1-Ellipse space, all concatenated into a single large vector $\boldsymbol{Z}$ - as the **Circle Gesture Language (CGL)**. It is this set of features that the `circle~` software package aims to recognize in incoming XY data.

3 SOFTWARE IMPLEMENTATION

The software is under active development - see Creighton (2026b) for a persistent record of the code used in this paper, or Creighton (2026a) for the up-to-date codebase. Supplementary files including video demonstrations are in Creighton (2026d).

3.1 Architecture

Figure 3 shows the overall architecture.

`circle~` is the main user-facing object. It receives raw XY data through two signal-rate connections, derives the high-D CGL feature vector, and outputs it as a multichannel signal from its first outlet. This object needs to be hosted in another patch - such standalone host patches are prefixed with `circleApp_`. For example `circleApp_Mouse` derives its XY data from the mouse position, `circleApp_tuioPad` listens for OSC data from the TuioPad app on a connected iPad, and `circleApp_featureTestbed` is a test environment in which the input to `circle~` is programatically generated. New users should start with the testbed, which provides an intuitive visual environment in which to explore the gesture language. We also provide `synth_` objects for synthesizing ellipses and 2-ellipse features, and `scope_` objects for visualizing the outputs from `circle~`.

The package has no dependencies apart from Max itself (with the exception of the `Rave20` demonstration, discussed below). It is hoped that `circle~` will have immediate "drop-in" usability for a wide range of Max environments due to its simple interface and low setup time.

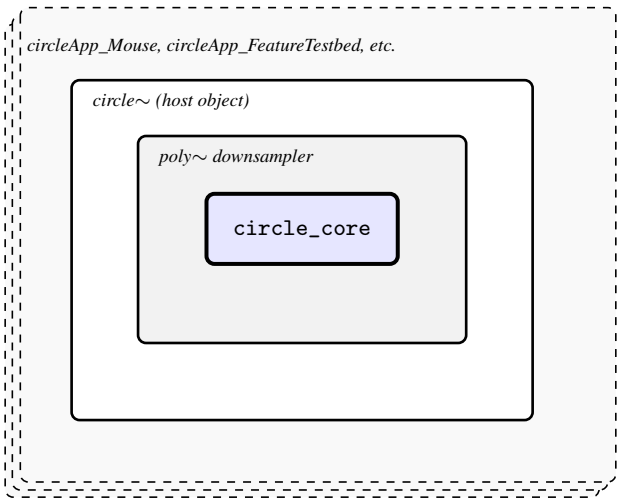


Figure 3: `circle~` architecture

Core signal processing is performed in `circle_core~`, which runs at a lower sample rate using Max's `poly~` object for downsampling.

3.2 Signal Processing

Figure 4 shows the signal flow through the core signal processing in `circle_core`.

3.2.1 Analytic Processing

The incoming $x[n]$ and $y[n]$ signals are passed through a filterbank to separate them into frequency bands. Then the clockwise and anticlockwise components are separated using an *analytic signal decomposition* based on the Hilbert transform. See Lyons (2010) and Oppenheim and Schafer (2010) for a detailed discussion of analytic signals and quadrature processing.

Once the clockwise $z^+[f,n]$ and anticlockwise $z^-[f,n]$ components at each frequency are known, the A and S components at each frequency can be found by eq. (8):

$$A[f,n] = \tfrac{1}{2}(z^+[f,n] + z^-[f,n]) \tag{25}$$

$$S[f,n] = \frac{(z^+[f,n] - z^-[f,n])}{(z^+[f,n] + z^-[f,n])} \tag{26}$$

3.2.2 PCA Tau Tracking

This block extracts τ using a PCA-based approach. For each (magnitude) frequency the filterbank outputs at that frequency are written into a 2D buffer - one could think of this as "drawing" the ellipse which is currently present in the signal at that frequency. By taking the principal components of this ellipse we measure the angle of its major axis with the x-axis and recover τ. See Johnson and Wichern (2007) for a discussion of PCA with respect to ellipse parameters.

This approach is "spatial" - that is, points in the 2D buffer are treated as a point cloud without ordering information. Each frame we remove the oldest sample in the buffer, replace it with the newest one, and calculate how that change changes the PCA components. Earlier attempts at a "temporal" approach by tracking relative phases were unsuccessful, since small errors in ψ or f cause a τ error which grows arbitrarily large over time.

3.2.3 Peak-Picking

This block looks for the k highest peaks in the amplitude spectrum (currently with $k = 4$), collects the corresponding f, S, τ and phase information and bundles it into `buf_ellipses` for further processing. The f estimate for each ellipse is improved by a sub-band parabolic interpolation as described in Smith and Serra (2005).

This peak-picking process works because we assume frequency sparsity in the incoming signal. Non-sparse inputs tend to activate many parameters at once in the feature vector output (which can have its musical uses).

3.2.4 Instantaneous Feature Extraction

This block has three stages.

First, it arranges the ellipses in amplitude order and compares their amplitudes to look for "one-ness", "pair-ness", "triple-ness", etc. It does this using a "group energy" measure:

$$G_k = \frac{k(A_k - A_{k+1})}{E_{total}} \tag{27}$$

where A_k refers to the k-th amplitude after sorting in descending amplitude order, and E_{total} is the "total energy" - i.e the sum of all amplitudes. A high G_1 indicates a strong concentration of energy into one ellipse, high G_2 suggests a pair of ellipses, etc. G_k can be viewed as an ensemble of OWA features (Yager, 1988) with weight vectors

$$W_1 = \frac{1}{E_{total}}[1, -1, 0, \ldots, 0] \tag{28}$$

$$W_2 = \frac{1}{E_{total}}[0, 2, -2, \ldots, 0] \tag{29}$$

$$W_3 = \frac{1}{E_{total}}[0, 0, 3, -3, \ldots, 0] \tag{30}$$

$$etc. \tag{31}$$

G_k can also be viewed as a "deconstructed L-statistic" - the calculation of the second L-moment and subsequently the Gini coefficient (Hosking, 1990) involves extracting an order statistic very similar to G_k before aggregating it into a global measure by summation.

We dwell on the links to previous literature because:

- The G_k decomposition was found autonomously by Google's Gemini LLM, using mathematics not previously known to the authors.
- We cautiously believe that this technique - taking an ensemble of individual measures rather than aggregating to a sum - is novel, though the general approach of sorting before application of a linear operator is not.

We discuss this this further in the Author Declarations under 'AI Ethics Statement'.

Having completed this calculation, the instantaneous frequency extractor uses G_k to determine how much "pair energy" is in each pair of ellipses from `buf_ellipses`. Consider a pair $P = \{A_{k_1}, A_{k_2}\}$ with $k_1 < k_2$. We define the pair energy E_P as the sum of all group energies G_k for which $k \geq k_2$, divided by the number of possible pairs in that group:

$$E_P = \sum_{k=k_2}^{k_{\max}} \frac{G_k}{\binom{k}{2}} \tag{32}$$

where $\binom{a}{b}$ denotes the binomial coefficient, or "a choose b".

For example - amplitudes A_2 and A_3 are both in the "top triple", so their pair energy receives a contribution from G_3, and similarly from G_4, G_5, etc. But they are not both members of the "top pair" or "top singleton", so G_2 and G_1 do not contribute. Energy from each G_k is equally distributed among the pairs in its corresponding pair/triple/quadruple/etc, hence the division by $\binom{k}{2}$.

Finally, the feature extractor bundles the information for each pair into `buf_instFeatures` for further processing. Each pair has an energy E, frequency information ω_0, ω_r, and shape parameters for the two ellipses. The shape parameters are transformed from the τ, S representation to the cartesian representation discussed in section 2.4.

3.2.5 Feature Smoothing

This block performs time-smoothing on the incoming information from `buf_ellipses` using a `lerp()` function, and outputs this information into `buf_smoothedFeatures`. After smoothing, `buf_smoothedFeatures` contains the best estimate of the $\boldsymbol{Z}$ coordinate vector according to `circle_core`$\sim$.

3.2.6 Output Stage

This block writes `buf_smoothedFeatures` and `buf_ellipses` to externally visible buffers for upsampling and output from `circle`$\sim$. The amplitude spectrum from the filterbanks is also passed on for use as a visual aid.

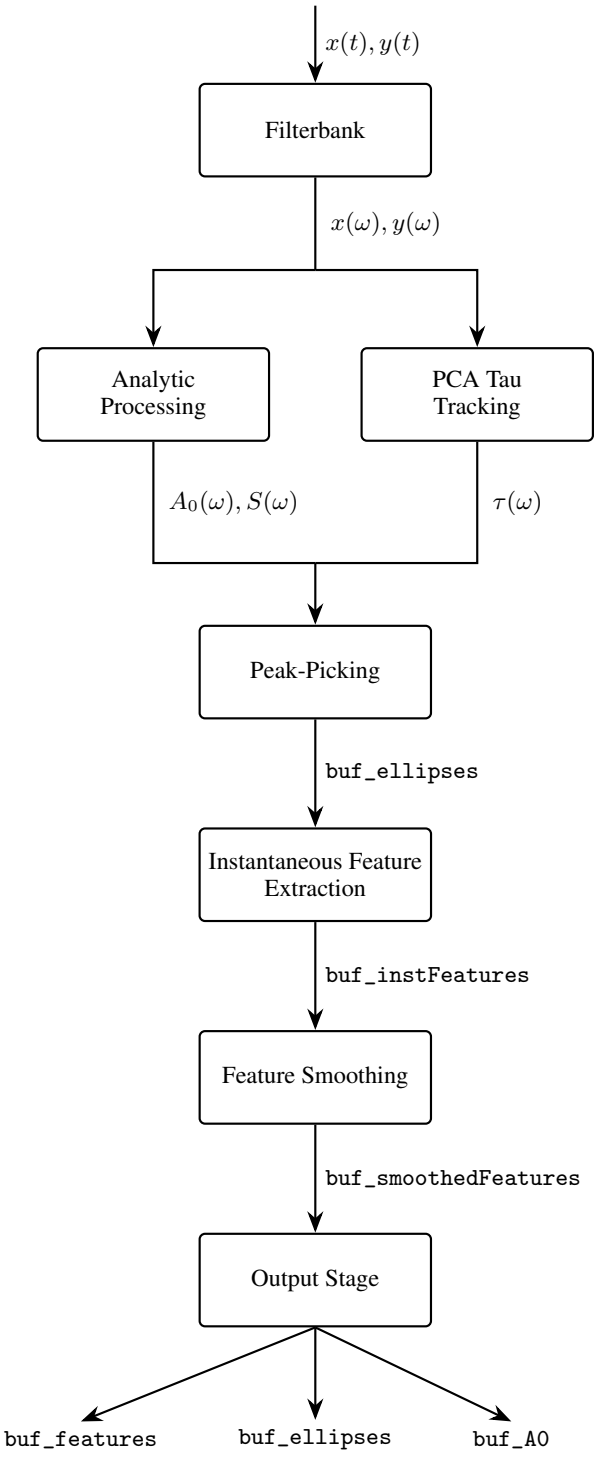


Figure 4: `circle_core` algorithmic block diagram.

4 EVALUATION

4.1 Software Verification

We provide a full-featured testbed in `circleApp_featureTestbed` to verify the operation of the package in a reproducible manner. The supplementary materials (Creighton, 2026d) include a video guide/demonstration of the testbed operation. we also provide a video demonstration with `circleApp_Mouse`, showing that the curves in the gesture language are human-drawable, and that the system can recognize curves on simple hardware in the presence of human errors, imperfect sensors, and noise.

4.1.1 Errors in ϕ

As these resources show, signal processing in `circle_core` tracks all values correctly, but for one - ϕ. As mentioned in section 3.2.2 'PCA Tau Tracking', small errors in frequency cause accumulating errors in phase measures, and no spatial-only method for extracting ϕ has yet been found. Values for ϕ have at best a large constant offset, and at worst are completely uncorrelated.

4.1.2 Audio Demonstration Patch

We provide a demonstration patch for how the tool might be used in the real world in `circleApp_Rave20`, with a video demo in the supplementary files. In this patch, we take advantage of the segmented nature of the feature space and control three instances of the RAVE VST with three separate ω_r subspaces of the CGL. In this demo the XY data stream for `circle`$\sim$ comes from the mouse for maximum interoperability.

We use E, the energy measure derived from G_k, to set the volume level of the RAVE instances, ensuring that they don't sound unless their corresponding feature is activated. The A_0 parameter is used a scaling parameter - larger shapes peturb the latent space vectors of the RAVE models further from their resting position. Since the 1-Ellipse space has smaller dimensionality than the other subspaces, we attach it to a model with fewer latent space dimensions. The total phase-space dimensionality of this demo is 26: $4+8+8 = 20$ dimensions for the latent space vectors, plus 3 dimensions for the A_0 scaling, plus 3 for the E scaling.

4.2 Critical Evaluation

The goals set in the introduction were: *navigable*, *high-D feature space*, *non-specialized hardware*, and *generic*.

The `circle`$\sim$ object runs well on a laptop, requires no dependencies other than Max, and can be controlled from a tablet or mouse, meeting the non-specialized hardware requirement. The feature space reaches 26+ human-controllable parameters comfortably, as shown in the video demonstration. as the authors and hopefully others practice with the tool we expect this number to rise. This meets the high-D feature space requirements.

The authors can at time of submission move smoothly between CGL curves, recall previous curves, and know which movements will produce which outcome. This meets the navigability requirement. It is worth noting that there are some areas of the CGL which the authors are still not able to traverse reliably: the $3/2$ subspace is, at time of submission the only ω_r-subspace with numerator larger than 1 which the authors navigate confidently in.

The question of "genericness" is more interesting. The topological properties of the $z_9(\cdot)$ feature space emerged over time - originally we aimed to design a highly euclidean controller for which all dimensions were independent and continuous, and all points reachable, with the view that this would lead to the most useful tool. As singularities and continuity constraints emerged we solved each problem as well as we could, trying to keep things as clean and rectangular as possible. Nevertheless we have ended up with a complex feature space with a non-trivial structure ... and it is that very structure which gives the tool its most interesting property, namely the ability to contain *multiple* separate subspaces controlling different tools.

It has, therefore, become less clear what form the tool "should" take - more topological complexity or less? Should the gesture language take a more unified or a more separated structure? What makes this type of tool useful in the first place? Revisiting and clarifying these design principles will be an important next stage in the development of the CGL.

4.2.1 Multiphonics, Transients and Smoothing Time

Two non-standard techniques have emerged through testing, practice, and undirected experimentation with the tool:

By drawing a curve with three or four frequency components, it is possible to activate multiple ω_r features simultaneously. These gestures are referred to as *multiphonics*, analogously to the wind-instrument technique. In a theoretical sense their existence is no surprise - the CGL describes a subset of the possible 2D curves and assigns feature vectors to those curves, remaining silent on curves outside that set. The software implementation has to assign *some* feature vector to *whatever* the user draws - so ambiguous curves end up with ambiguous feature vectors.

In practice, multiphonics are interesting because they allow us to transcend some of the limits on simultaneous nonzero parameters, allowing for more flexible control.

Short, jerky, "transient" movements produce fluctuating values across many features. Movements which are localized in time are nonlocal in frequency - and therefore they often contribute to several ω_r features at once. They also tend to confound the peak-picking logic in `circle_core` which assumes sparsity in its input signal. The combined result is that energy jumps between features after a sharp movement, as the pick-picking logic changes its choice of peaks erratically, and this creates a multi-band fluctuating effect.

Another way to transcend the simultaneous-nonzero limitations is simply to raise the time constant of the smoothing block, such that previously drawn features persist as new ones are drawn. Further work on this system is planned to include an expression pedal which can change the smoothing time in real time.

5 CONCLUSIONS

We described the Circle Gesture Language (CGL) - a new tool for expressing high-dimensional control actions on low dimensional controllers in real time. We tested the new system from numerical and user-focused points of view, and reflected on its capabilities.

The prominent remaining technical question is - how does the unique topology of the gesture language map to the

topology of latent spaces in the wild? Can CGL be used for very high dimensional control in a single latent space as originally envisaged, or is it best suited to lower dimensional control across multiple spaces at once? To shed light on this, we plan to connect the CGL to a larger range of models and spaces, including concatenative (Thibault et al., 2025) and differentiable wavetable (Shan et al., 2022) paradigms, testing the system in both single-model and multi-model configurations. Work will continue to understand the CGL topology and reformulate it if necessary.

Investigations are ongoing into the artistic potential of the system - the authors are preparing a 1 hour performance demonstrating the use of CGL as a control paradigm in both solo and ensemble settings. Another key goal is the creation of a small self-contained tool and its distribution for user studies and design feedback.

AUTHOR DECLARATIONS

AI Ethics Statement

Google's Gemini LLM was used extensively during this project, for example to to reformulate and invert systems of equations, reason about bugs and new features, and discover existing research and resources.

As discussed in section 3.2.4, Gemini at one point described an OWA feature which may be novel. Gemini received no prompting from the user to use OWA features or L-statistics in its solution, and in fact a polynomial approach had been suggested by the researcher in question. Chat logs are provided in Creighton (2026c).

According to AIMC's authorship guidelines in Committee on Publication Ethics (2023) and Association for Computing Machinery (2025), AI tools have no authorship rights since they cannot meet point 3: "Authors must be accountable for the work that was done and its presentation in a publication". Further, the authorship of the current set of authors is not in question as there are several other claims to novelty in the paper and all authors made a substantial contribution to the work. All novel work not discussed in this section was contributed by a human author.

The new OWA feature underwent the same evaluation and verification process as any other part of the system. This included rederivation by hand, review of existing literature, unit testing, full system testing, demonstration in a user-facing setting, and, ultimately, submission for peer review.

It is the opinion of the authors that scrupulous honesty and disclosure is more important than ever as the epistemological environment changes around us. Without an accurate read on when and how researchers are using LLMs and what contributions they are making, the community will struggle to adapt.

Conflicts of Interest

Author M. Creighton is an inventor on a pending UK patent application related to the mathematics described in this paper, with Patent Application Number GB2613566.5. No other conflicts of interest are declared.

REFERENCES

Arfib, D., Couturier, J. M., Kessous, L., and Verfaille, V. (2002). Strategies of mapping between gesture data and synthesis model parameters using perceptual spaces. *Organised Sound*, 7(2):127–144.

Association for Computing Machinery (2025). ACM Policy on Authorship. https://www.acm.org/publications/policies/new-acm-policy-on-authorship. Updated September 16, 2025; Accessed May 3, 2026.

Caillon, A. and Esling, P. (2021). RAVE: A variational autoencoder for fast and high-quality neural audio synthesis. *CoRR*, abs/2111.05011. arXiv: 2111.05011.

Caspe, F., McPherson, A., and Sandler, M. (2025). Waveform autoencoding at the edge of perceivable latency. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 73–76, Canberra, Australia. Publisher: Zenodo.

Choi, I. (1998). A Manifold Interface for Kinesthetic Notation in High-Dimensional Systems. In Wanderley, M. M. and Popovic, D., editors, *Trends in Gestural Control of Music*, pages 9–26. IRCAM.

Committee on Publication Ethics (2023). Authorship and AI tools. Technical report, COPE.

Creighton, M. (2026a). Circle: Main repository. https://github.com/MattCreightonAudio/circle/.

Creighton, M. (2026b). Code snapshot relating to "circle: high- dimensional latent space traversal in real time on a two-dimensional controller", aimc 2026. https://doi.org/10.5281/zenodo.21939827, https://github.com/MattCreightonAudio/circle/releases/tag/AIMC2026.

Creighton, M. (2026c). Llm chat logs relating to "circle: high- dimensional latent space traversal in real time on a two-dimensional controller", aimc 2026. https://doi.org/10.5281/zenodo.21957578.

Creighton, M. (2026d). Supplementary files relating to "circle: high- dimensional latent space traversal in real time on a two-dimensional controller", aimc 2026. https://doi.org/10.5281/zenodo.21957045.

Engel, J., Hantrakul, L., Gu, C., and Roberts, A. (2020). DDSP: Differentiable Digital Signal Processing. arXiv:2001.04643 [cs].

Fiebrink, R., Cook, P., and Trueman, D. (2009). Play-Along Mapping of Musical Controllers.

Fiebrink, R. and Cook, P. R. (2010). The Wekinator: a system for real-time, interactive machine learning in music. In *Proceedings of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)(Utrecht)*, volume 3, pages 2–1.

Hosking, J. R. (1990). L-moments: analysis and estimation of distributions using linear combinations of order statistics. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 52(1):105–124.

Johnson, R. A. and Wichern, D. W. (2007). *Applied Multivariate Statistical Analysis*. Pearson Prentice Hall, 6th edition.

Lyons, R. G. (2010). Understanding digital signal processing. In *Understanding Digital Signal Processing*, pages 439–499. Pearson Education, 3rd edition. Chapters 8 and 9.

Martelloni, A. and Kiefer, C. (2025). Towards an Ecosystem of Instruments of Tunable Machine Learning. Publisher: Zenodo.

Mulder, A. (1998). Virtual Musical Instruments: Accessing the Sound Synthesis Universe as a Performer.

Oppenheim, A. V. and Schafer, R. W. (2010). Discrete-time signal processing. In *Discrete-Time Signal Processing*, pages 793–844. Pearson, 3rd edition. Chapter 11: Discrete Hilbert Transforms.

Peachey, M., Oore, S., and Malloch, J. (2025). Evaluating Low-Dimensional Latent Representations as a Creative Interface for Digital Synthesizers. Publisher: Zenodo.

Privato, N., Shepardson, V., Lepri, G., and Magnusson, T. (2024). Stacco: Exploring the Embodied Perception of Latent Representations in Neural Synthesis. Publisher: Zenodo.

Shan, S., Hantrakul, L., Chen, J., Avent, M., and Trevelyan, D. (2022). Differentiable Wavetable Synthesis. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4598–4602, Singapore, Singapore. IEEE.

Smith, J. and Serra, X. (2005). PARSHL: An Analysis/Synthesis Program for Non-Harmonic Sounds Based on a Sinusoidal Representation. *Proceedings of the International Computer Music Conference*.

Tahiroğlu, K., Kastemaa, M., and Koli, O. (2021). AI-terity 2.0: An Autonomous NIME Featuring GANSpaceSynth Deep Learning Model. In *NIME 2021*, Shanghai, China. PubPub.

Thibault, D., Piazza, D., Allard, M., and Arseneault, M. (2025). Navigating Abstract Timbre Spaces: Instrumental Affordances of Concatenative Synthesis in Mosaïque. Publisher: Zenodo.

Yager, R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):183–190.

Zheng, S. J., Xambó Sedó, A., and Bryan-Kinns, N. (2025). Exploring gestural affordances in audio latent space navigation. *Frontiers in Computer Science*, 7:1575202.