Topographic Synthesis: Parameter Distribution in Spatial Texture

Erik Nyström
University of Birmingham, UK
e.p.nystrom@bham.ac.uk

ABSTRACT

Topographic synthesis involves the distribution of arrayed structures of parameter values for simultaneous synthesis processes assigned to different channels in a multi-loudspeaker system. In this model, the concept of sound synthesis extends to the design of spatial texture, and morphology is considered not only as change in sound over time, but also as an instantaneous difference in spatially distributed simultaneous sound. The term topography, here, refers to the sonic relief articulated across the perspectival field – the array of possible sonic spatial perspectives within a loudspeaker system – as spatial configurations are shifted over time. The paper presents a set of algorithms which distribute relative properties of texture in multichannel speaker arrays, especially relevant to high density loudspeaker arrays (HDLA), some of which have non-linear, self-organising properties. The processes have been designed in SuperCollider, for a flexible live context, but can be applied in fixed media composition as well.

1. INTRODUCTION

Much research on spatial audio in computer music is focused either on the infrastructure of sound systems, or on the positioning, moving, or spreading of images in space in a precise and robust manner. Related composition and performance approaches are often based on taking sounds or pieces created or recorded in stereo or multichannel formats and magnifying or positioning (decoding) them within an array of channels, to spatially reproduce the sound in the most ideal way. The field of spatial possibilities is treated as neutral in the sense that the task of the spatial distribution is to accurately project a sound image. This is concurrent with an illusionistic approach to image and spatial perspective, which is about creating a life-like spatial scene which obscures the presence of technology. In the approach presented here, however, the sound design process begins with the spatial distribution, and the resulting spatial texture is entirely a synthesis of interaction between loudspeaker channels. The type of spatiality created here is not easily classified in terms of, for instance, distant or proximate material, but is rather more like an emerging web, suspended at the peripheries of the listening space, and projecting across the listener. Because of this distributed synthesis approach, composing spatial behaviour is the same as composing morphology. The absence of this space-sound duality is the reason why this paper will avoid classifying the approaches as 'spatialisation' – since this term seems to imply that space is an added rather than intrinsic dimension of sound. Instead, the term *topographic synthesis* is used to imply the composition of spatially differentiated sound fields. The approaches described here broadly share the philosophy outlined by Kerry Hagan [1] in that they are not about what she refers to as "mimetic spatialisation, that is, the mimicry of actual sounds in space" (p. 36) and they are largely decentralised, circumspatial[1], and based on point-source spatial sound production (sound assigned to discrete speaker channels). Recent examples of others who have used approaches to spatial distribution as basis for structure and material include Rama Gottfried [2] who experimented with the artefacts of wave field synthesis and high-order ambisonics algorithms. This bypasses the encode/decode duality present in much multichannel music, where a definitive version of a sound or work is adapted to a format, as instead the system itself is fundamentally linked to the agency of composition.

This paper presents some tools developed in SuperCollider[2] [3, 4] for composing by distributing parameters across synthesis processes discretely assigned to channels in a loudspeaker array[3]. While the interface of some of the processes loosely favour certain structures of loudspeaker configuration, most of them are agnostic to spatial layouts, so can be used creatively for a variety of results, and will certainly have possible uses which this author has not yet considered. Technical structures for distribution of parameters will be outlined in context of spatial music composition and performance, with critical observations on their use in practice. One important contextual consideration is that the tools were developed for use in live performance, and for synthesising spatial sound in real-time. The interfaces have therefore been developed with the idea that one parameter, or just one iteration of an algorithm, should be able to generate a multitude of different parameter values for multichannel distribution. The motivation for this is practical in that a composer/performer accesses many channels in one action, but also aesthetic, in that formalisation of spatial structures are made possible, and in that some degree of technological extension of, and interaction with, human control is created.

[1] Circumspace is a term used by Denis Smalley, defined as 'perspectives that encompass the listener' [9, p. 51].
[2] https://supercollider.github.io
[3] https://github.com/postnature/parameter-distribution

2. INTEGRATION AND SEGREGATION

The approach discussed here has in common with spatial 'decorrelation' techniques – such as the application of granular processing or phase-vocoding in order to differentiate sound among multiple channels – that each source channel contributes with a *different* signal towards a *unified* sound or texture. Gary Kendall [5] has cited several reasons why decorrelation is effective for spatial distribution, and more robust than panning models. Among these, some important points are that they can create a diffuse contiguous field, they avoid 'image shift' – the warping of an image relative to position of listener – and they eliminate the precedence effect, which causes the nearest speaker to determine location if the same sound is projected across many speakers [5, 6, 7]. The term 'decorrelation' is, however, not entirely appropriate for the present approaches, since it implies imparting a process onto audio to increase the quantity of differentiated channels. Here, on the contrary, we begin at the different channels, and synthesise a texture by *integrating* and *segregating* sonic properties by structuring relative parameter data. A strongly integrated texture has very similar output from different channels, resulting in a strong perceptual fusion, and a unified texture. In a very segregated texture, on the other hand, one can hear more clearly the loudspeakers as spatial sources, because the signals in the channels are have more difference from one another. Temporal properties have a stronger capacity for perspectival differentiation than spectral properties, because transients are easily localised. Spectral difference, on the other hand, does not create as strong spatial gaps between channels, and can appear to create an interpolated spatial shape.

A key aspect of listening here is the liminal territories between 'one' and 'many' (textures/sounds). Kendall [8] has described spatial attributes in terms of schemata, and foregrounded how the gradient territory between SOURCE – a single point in space – and ENSEMBLE – a collection of sounds – can be a powerful resource of play with perceptual grouping in electroacoustic music. Spatial texture is often manifest as a single canvas within which we are able to hear multiple localities. Thus, integration and segregation are important forces of spatial tension and cohesion.

3. INSTANTANEOUS AND EVOLUTIVE STRUCTURES

Spatially distributed texture can be articulated in two primary 'modes' of space-time. A distinct aspect of spatial texture is that it always has *instantaneous* structure in space, in that it features spatial relationships which are not time-dependent. As an example, imagine a texture which has a spectral tilt across horizontal space, so one area occupies a high frequency range, and then a sloping frequency range is created towards the opposite end. This structure is instantaneous, because the different spectral parts of the texture are simultaneously present. Regardless of whether listeners consider spectrum in itself to be a spatial dimension (as in Denis Smalley's term *spectral space* [9]), spectral differences distributed across the perspectival field does create a spatially extended structure.

On the other hand, spatial texture may also create shapes which would not have appeared were it not for the perception of difference over time. If, for instance, the spectral tilt were to shift around the listener in a circular manner, an *evolutive* circle would be created. Thus, evolutive space is linked to the more traditional conception of sound shape, or musical form(s) as temporal phenomena, wheras instantaneous space is structured by non-temporal dimensions. The difference between the two is important for the present discussion because it helps understand the technical procedures described, which, in turn, are motivated by the aesthetic concern with composing topography. To simplify, we can say that, in spatial texture, evolutive processes interpolate between instantaneous spatial configurations.

4. SYNTHESIS METHODLOGY

The general method described here has the following steps:

1) Design a single-channel synthesis process.
2) Assign different instances of this synthesis process to different channels in a multichannel loudspeaker configuration.
3) Set synthesis parameters across all channels to articulate spatial texture.

In step two, a mono signal should appear if the synth in question has no indeterministic features and the parameter settings of all instances are identical. If, however, the signal is noise-based, or has built-in irregular modulations, a uniform but spatially spread texture will be perceivable, extending to all loudspeakers used. The third step is where spatial synthesis happens through distributing parameter settings in space and time.

4.1 Useful Synthesis Parameters

A basic observation in this context is that one needs consider, on one hand, the sonic properties which engender sufficient similarity across a spatial field so that the mass of sound can be considered a texture; and, on the other hand, what the variability of these properties are for differentiation. We can use as a rule of thumb, that microtemporal deviations (among for instance particles or modulations) among channels cause strong spatial segregation within a texture, in that the texture will be anchored in the loudspeaker sources. Conversely, the more similar, synchronic, or coinciding, micro-level properties are, the more the texture will integrate. Spectral differences will create a more fused perspectival field, in that locations will not be articulated, but distributed difference is clearly perceived. Of course, harmonic relationships will promote spectral integration. The combination of spectrum and micro-time is essential to forces of integration and segregation. Patterning also occurs on higher timescales and have significant impact on motion, but micro-time relationships will always be important to the spread of texture in any situation.

Any general synthesis techniques can be used in the topographic model, but for it to work well, some general guidelines are useful, regardless of the nature of the specific synthesis algorithms used. Firstly, if the process is able to establish steady frequencies/pitches (e.g. through periodic oscillators) or focus around spectral centres (e.g. filtered noise), then that means that spectral difference and similarity can be used effectively for spatial articulation across the texture. If the process also can do the opposite – i.e. very unstable behaviour – it is yet more powerful, since a range between unison and disorder will be possible. Additional possibilities for spectral articulation are also useful, simple examples could be high-or low-pass filters that can shape the spectrum across the texture even if there is a uniform fundamental pitch or similar spectral centre. More complex examples could be control of partials in an additive spectrum, or control of modulations in FM synthesis. Secondly, parameters for micro-temporal articulation are essential if any kind of resolution of locality is desired. Thus, if some variety of particle synthesis [10] is used, any of the standard granular synthesis parameters are welcome, e.g., grain envelopes, durations, rate, periodicity. If, on the other hand, the synth is producing a smooth, continuous sound (e.g. from an aperiodic or periodic oscillator) it is good have the optional introduction of amplitude or frequency modulation available, for instance, through oscillators or triggered envelopes, so that a surface complexion can be introduced. For any particle-generating process or modulation, continuous variability of periodicity is effective, since a unison of periodic modulations integrates a texture, whereas the decorrelation of aperiodic impulses among channels engenders spatial segregation. Non-standard synthesis models, such as GENDYN [11], Nick Collins' SLUGens [12], or other non-linear synthesis models, are very powerful for spatial texture due to their timbral ranges.

These points are relevant to psychoacoustic factors in spatial hearing, where micro-temporal differences have an important influence on localisation, due to the precedence effect [6]. Though we are not especially interested in localising individual sounds here, the coincidences and differences of micro-time properties can create both radical shifts between texture integration and segregation, and subtle 'phasic' texture motion, where the relations among coinciding pulsations in the texture create a 'rippling' spatial effect [13]. Spectral differences are less distinct as localising cues, which means that one can spatially distribute the timbre of the texture without introducing noticeable gaps. Possible musical uses include harmonic relationships, beat frequencies, difference tones etc. Spectral properties also affect the impression of verticality in texture and is important for stratification [14].

5. PARAMETER DISTRIBUTION

Parameter distributions are formalisations of instantaneous spatial structure. Distributions are applied either to sustaining/continual synthesis processes, as updates of settings, or to new events in time (e.g. grains), each of which simultaneously occupies multiple channels. The following sections explain SuperCollider classes which output arrays of values, to be mapped to parameters of spatially distributed synths. The reader is referred to https://github.com/postnature/parameter-distribution for code and practical examples.

5.1 Instantaneous Envelopes

Parameter distribution envelopes offer the most predictable and intuitive approach to the composition of instantaneous structures. They generate arrays of values which are equivalent to breakpoints in an envelope, each breakpoint representing a channel. In combination with temporal interpolation, this can be a very powerful method for creating dynamics in spatial texture, by biasing textural properties towards different directions in the perspectival field, or altering the curvature of the envelopes. How the envelope shape is perceived depends strongly on what kind of parameter it is mapped to. If it is mapped to the centre frequency of a bandpass filter fed with noise, the spectral gradient will be rather smooth, since there is spectral overlap between channels. If it is mapped to oscillator frequencies the individual frequencies will form a spatial timbre of distributed partial frequencies.

The class `ParamPeak` creates a peak envelope with equal slopes on either side. By specifying a peak position, the apex of the envelope can be located at a channel in the speaker array. Depending on how wide the envelope is, values will decrease for every channel further away from the peak. The interface, programmed for an eight-channel distribution of frequency values, could look thus:

```
ParamPeak.new(numPoints: 8, peakWidth: 6, peak-
Pos: 3, minVal: 3000, maxVal: 6000, curve: 2)
```

This generates the array illustrated in figure 1, where breakpoints represent channels, which could be physically arranged in any configuration, for instance a circle.

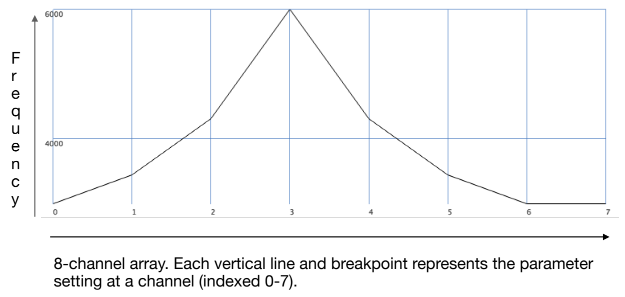

Figure 1 Envelope illustration of the array[3000, 3445, 4312, 6000, 4312, 3445, 3000, 3000].

`ParamPeakWarp` also creates a peak envelope but warps depending on where the peak position is in relation to the edges of the array. Thus, it is not symmetrical unless the peak is at the centre of the array. The below setting is illustrated in figure 2.

```
ParamPeakWarp.new(numPoints: 8, peakPos: 3, min-
Val: 3000, maxVal: 6000, curve:2)
```

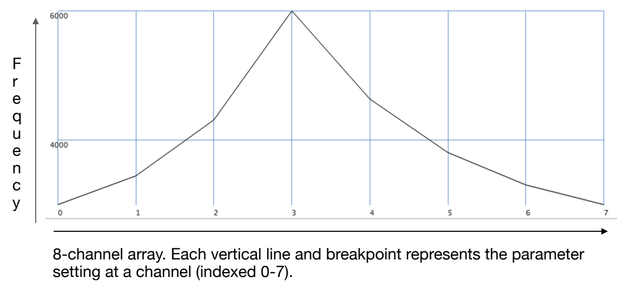

Figure 2: Envelope illustration of the array[3000, 3445, 4312, 6000, 4635, 3807, 3305, 3000].

Shifted to one side, however, it produces results as in figure 3.

```
ParamPeakWarp.new(numPoints: 8, peakPos: 1, min-
Val: 3000, maxVal: 6000, curve:2)
```

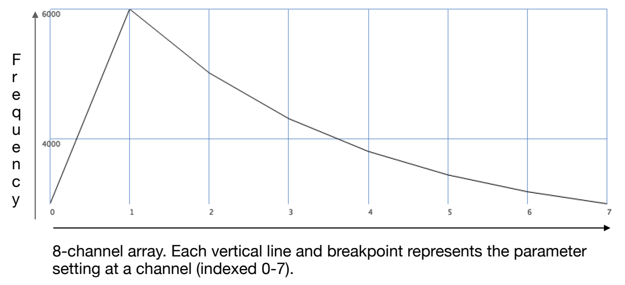

Figure 3: Envelope illustration of [3000, 6000, 5016, 4312, 3807, 3445, 3186, 3000].

If the peak position of the above envelopes is moved continuously over time a sort of timbral panning will be achieved.

ParamCurve creates a single curve from a minimum to maximum value. This distribution is powerful when its instances are interpolated and mapped to spectral or microtime properties, resulting in a continuous motion between integrated (flat curve) and segregated states (steep curve), so the texture expands and contracts. Figure 4 illustrates,

```
ParamCurve.new(numPoints: 8, minVal: 3000,
maxVal: 6000, curve: 2)
```

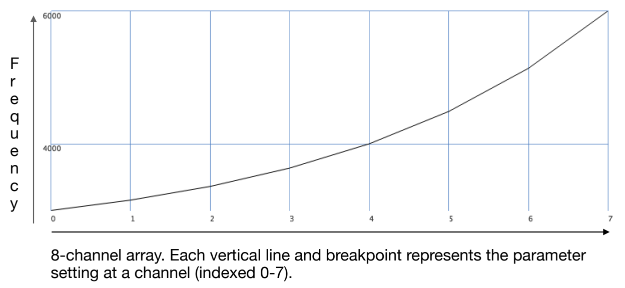

Figure 4: Envelope illustration of [3000, 3155, 3361, 3637, 4002, 4489, 5137, 6000].

ParamLatPairs is an extension of ParamCurve specially designed for a pair-wise channel ordering, imagining a system with front and rear. This front-to-rear dimension is referred to as longitude, whereas the sideways dimension is latitude [13, 14]. The interface specifies how many lateral pairs there are across the longitude. E.g in an 8-channel setup that would be four left-right pairs: 1-2, 3-4, 5-6, and 7-8. The output array thus follows this arrangement. A curve is created from front to rear, a lateral tilt is applied so that a left-right shift can be introduced. The tilt is a positive or negative value within the range +-1, which determines a skew in left or right direction, and can increase or decrease along the longitude with latTiltF for tilt at the front-most pair, and latTiltR for tilt at the rear-most pair. The widthComp argument is an array of the same size as numLatitudes, which determines a compensation for the width of loudspeaker pairs. E.g. in a pairwise eight-channel configuration, 1-2 and 7-8 may be narrower than 3-4 and 5-6, and thus the array for widthComp could be [0.5, 1, 1, 0.5]: the values on left and right will then be adjusted for width so that the tilt angle is more similar. The array output follows the pairwise ordering. Thus,

```
ParamLatPairs.new(numLatitudes: 4, frontVal:
0.1, rearVal: 1.0, curveLong: 4, latTiltF: 0.0,l
atTiltR:0.3, widthComp: nil)
```

generates: [0.1, 0.1, 0.145, 0.149, 0.301, 0.351, 0.7, 1.429]. In this example, the tilt increases from none at the front (0.1, 0.1) to an up-right weighed skew at the rear (0.7, 1.4285714285714). With widthComp at [0.5, 1, 1, 0.5] instead of nil, we get a lesser skew at the rear (the front is unaffected since there is no tilt there anyway): [0.1, 0.1, 0.145, 0.149, 0.301, 0.351, 0.85, 1.176].

ParamField, a more complex application of the same concept, creates a latitude/longitude distribution for a grid-wise layout of loudspeakers, for instance an array of 16 suspended speakers in 4 by 4 arrangement. ParamField is mathematically different from ParamLatPairs only in that it has a lateral curvature along which the multiple lateral points are scaled. Thus,

```
ParamField.new(numLatitudes:4,numLongitudes:4,
frontVal:1000, rearVal:4000, curveLong:4, lat-
TiltF:0.1, latTiltR:0.2, curveLat:0,
latCurveWarp:0)
```

results in: [900, 970, 1041, 1111, 1035, 1121, 1206, 1292, 1531, 1687, 1843, 1999, 3200, 3800, 4400, 5000].

ParamDeviation is a simple generator of multichannel stochastic deviation from a given mean. It is easily integrated with envelopes through the scaleArray argument, where an instance of e.g., ParamCurve, can be used to create a slope. For instance, the setting,

```
ParamDeviation.new(numPoints:8, val:1000, devia-
tion:0.1, scaleArray: ParamCurve(8,1,0.8,\exp),
dist: \gauss, boundary:\clip, minVal:500,
maxVal:2000)
```

produces a stochastic distribution of values with a slight slope, such as, [1185, 1001, 809, 886, 886, 941, 784, 791]. ParamDeviation can also be used effectively to spread a texture between vertically distributed channels. A simplified example could be a texture spread over two channels, one above the other. If ParamDeviation is used to generate frequencies independently for each channel, and is set to have a scalearray of [1, 2] then the second channel will be on average an octave above the first, which creates a vertically manifest spectral structure.

5.2 Non-linear Distribution Processes

The parameter envelopes above are linear, in that the same setting always produces the same result (or statistically the same, in the case of ParamDeviation). The non-linear distributions, on the contrary, are manifest as spatiotemporal processes where the output values in each channel are affected by one another, and by their previous states. The correspondence between setting and generated values depends on the current state, which changes over the course of iterations. The influence that emergent history and neighbouring states have on each channel means that these distributions can generate new values without requiring a new input at each iteration, and can display self-organising instantaneous and evolutive structures which are closely linked, not unlike cellular automata.

ParamFeed is based on the idea of feeding data instantaneously through the channels, and back from each channel to itself over successive iterations. It has options for 'mixing' the addition of parameter settings from previous state, left neighbour, and current user input setting, creating a form of data feedback system which wraps around the array of outputs. It was developed with the aim of creating a model which will generate an array of different parameters with only one input value, in a causal but non-linear manner. In performance, this allows for all channels of a texture to be controlled by a single knob or fader with a non-linear response. The behaviour of the class can be updated with setter methods which might be mapped to additional controllers in performance. The values fold when they go outside the range specified, and this aliasing can introduce unexpected behaviours in the response. This algorithm is instantiated with a number of spatial points (channels), minimum and maximum values, and multipliers for how much of the input value to add to each cell, how much of the left neighbour, and how much of the previous value in each cell to add to the new one. If at least one of these arguments is 1 or more the array of values will inevitably increase on every iteration; if all is less than 1, the values will decrease. Example of an instance:

```
x=ParamFeed.new(numPoints: 8, minVal: 0.1,
maxVal:1.0, inValMul: 1, accumFeedMul: 0.1,
prevFeedMul: 0.1, deviation: 0.0)
```

Calling .next on it will give an array of new values for each point in the array. Thus, executing x.next(inVal: 0.3), yields the smooth curve [0.3, 0.33, 0.363, 0.399, 0.439, 0.483, 0.531, 0.585]. Executing the exact same line of code again will generate a different result.

If setting inValMul to 0 instead and accumFeedMul to 1, the new distribution will build primarily on what was in the left neighbour and the input value will not be counted. Thus a more irregular result [0.833, 0.33, 0.205, 0.414, 0.834, 0.325, 0.207, 0.422] is generated.

The deviation argument introduces a scattering around the values generated by the distribution. Calling the method .nextCell instead of .next will generate just one value for a given index in the array of channels, rather than all at once, though the full array is still updated.

ParamCells and ParamCellFunc are a form of cellular automata where parameters are generated along a continuous scale. ParamCells implements a basic continuous form of cellular automata, where the user can control the minimum and maximum boundaries of the parameter space and feed any value into a chosen index in the array (if desired). On every iteration, each channel creates an average in relation to its left and right neighbours, according to a user supplied or random left/right weighting (neighbourBiasFunc). If the warp argument is set to less 1, the output values gravitate towards the centre between minVal and maxVal, and if set to more than 1, values will tend outwards. The curve argument scales the warp as appropriate for the parameter in question (e.g. exponential for frequency). errorProb is the probability for a random replacement of a cell's value. The interface looks as such:

```
x=ParamCells.new(numPoints:8,    minVal:    2000,
maxVal: 12000, curve: \exp, errorProb: 0.1,
neighbourBiasFunc: {1.0.rand} ,warp: 1)
```

Executing x.next will generate an array of values within minVal and maxVal. Calling the same method with arguments specified, e.g. x.next(inVal: 4000, index: 3), will influence the outcome by replacing index 3 with the value 4000 before calculating the output array.

ParamCellFunc allows for specifying the rules of interaction between cells within a user-supplied function. On every consecutive iteration, each cell will update itself by executing the function, which is supplied the value of the cell and its neighbours.

The successive outputs of the cellular algorithms jump abruptly from a value to another, creating 'fractional motion', where states in the texture alter in an abrupt fashion [13].

5.3 SuperCollider Patterns

The Param classes are also implemented as SuperCollider patterns (named Pparam), integrating with the existing algorithms for composition and parameter manipulation available in the SuperCollider pattern library [15]. The Pparam interfaces are almost identical to the Param ones, and they also output arrays, which means that an enclosing Pbind or Pmono should be multichannel expanded to generate an array of synths. Notably, using the patterns allows for the possibility to interpolate between values generated over time using the pattern Pseg. Thus, PparamPeak can be used to pan a parameter peak around a system; PparamCurve can be made to behave in an elastic manner; and the otherwise discontinuous output of PparamCells, and PparamDeviation can be interpolated for smoother texture motion.

6. DISCUSSION

The tools have been used in composition – e.g. in the author's works *Spheroid* and *Texton Mirrors* – and performance on different HDLAs including BEAST (Birmingham Electro-Acoustic Sound Theatre, University of Birmingham, UK), The Cube at Moss Art Centre, ICAT (Institute for Creativity, Arts and Technology, Virginia Tech, Blacksburg, VA, US), and The Cube at IEM (Institute of Electronic Music, Graz, Austria). The tightly arranged 64 channel array of speakers in The Cube at VT was ideal for sweeping a `ParamPeak` mapped to frequency and distortion of a simple oscillator synth across the room, creating a timbral wave. This distribution, however, does produce audible 'bumps' when sweeping on more regular systems where speakers are a few metres apart. That said, the sweeping motion is only one way of using `ParamPeak` and not necessarily one that is well suited to a textural aesthetic, since it produces a gestural trajectory within the field. One can also work mainly with the height of the peak, and move the position only when the peak is low, so that the sweep is disguised. `ParamCurve` works well in most types of speaker setup, although the envelope curvature will be significantly more pronounced, the more channels are available. Note, however, that the parametric curve can be in conflict with the architecture of the sound system. For instance, if a `ParamCurve` array were to wrap around a rectangular system (as in The Cube at VT), the physical corners may be in tension with the smooth curve of the texture. Therefore, mapping a curve along one side, or in circular arrays (such as those within BEAST) can make more sense. `ParamField` was tested in the top ceiling grid of 20 speakers in The Cube at VT (mapped to frequency), and could create an elastic, 'crawling' texture motion due to interpolations between different curvatures and tilts. A similar result was yielded for the ceiling grid at IEM's Cube in Graz.

The cellular processes create a behaviour where the values in each channel regulate one another so that an irregular global pattern of fluctuation between minimum and maximum is created. The sounding outcome is entirely dependent on what parameters the distribution is mapped to. Mapping a MIDI knob so that each new cc value generates a new iteration and array of frequencies enables a mode of performance where minute bodily interaction with control generates significant spatial shifts, but also offers the option of staying at a state for any duration of time. This sensitivity stimulates performance in an interesting way and produces a texture of abruptly switching perspectives.

In the author's practice, these tools have been used primarily for distributions of spectral properties such as frequency, bandwidth or distortion, but also for local micro time parameters such as modulation envelopes. This is often done in conjunction with temporal modulations or impulse streams whose rates are global to the whole texture (or grouped into spatial zones), but change in synchrony with spectral alterations, and create an integrating tension to the texture as a whole. In these cases, topographic synthesis creates a cohesive spatial field whose perspectives shift abruptly or gradually over time, and which resists collapsing due to precedence effect, since the signal is different in all channels.

Acknowledgments
The research was funded by a Leverhulme Early Career Fellowship at BEAST, University of Birmingham, UK. Thanks to Eric Lyon for research time in The Cube at ICAT in August 2016, and Marko Ciciliani for time in The Cube at the IEM in Graz in April 2018.

7. REFERENCES

[1] K. Hagan, "Textural Composition: Aesthetics, Techniques, and Spatialization for High-Density Loudspeaker Arrays.," *Computer Music Journal,* vol. 41, no. 1, pp. 34-45, 2017.

[2] T. Carpentier et al, "Holophonic Sound in IRCAM's Concert Hall: Technological and Aesthetic Practices," *Computer Music Journal,* vol. 40, no. 4, pp. 14-34, 2016.

[3] J. McCartney, "Rethinking the Computer Music Language: SuperCollider." *Computer Music Journal* vol. 26, no. 4, pp. 61-68, 2002.

[4] S. Wilson, D. Cottle, and N. Collins, Eds., *The SuperCollider Book*. Cambridge, MA: MIT Press, 2011.

[5] G. Kendall, "The Decorrelation of Audio Signals and its Impact on Spatial Imagery," *Computer Music Journal,* vol. 19, no. 4, 1995.

[6] J. Blauert, "Localisation and the Law of The First Wavefornt in the Median Plane," *Journal of Acoustical Society of America,* vol. 50, no. 2, pp. 466-70, 1971.

[7] H. Haas, "On the influence of a single echo on the intelligibility of speech," *Acoustica,* vol. 1, pp. 49-58, 1951.

[8] G. Kendall, "Spatial Perception and Cognition in Multichannel Audio for Electroacoustic Music," *Organised Sound,* vol. 15, no. 3, pp. 228-238, 2010.

[9] D. Smalley, "Space-Form and the Acousmatic Image," *Organised Sound,* vol. 12, no. 2, pp. 35-58, 2007.

[10] C. Roads, *Microsound*, Cambridge: MIT Press, 2001.

[11] I. Xenakis, *Formalized Music*, New York: Pendragon Press, 1992.

[12] N. Collins, "Errant Sound Synthesis," in *Proceedings of ICMC*, 2008.

[13] E. Nyström, "Morphology of the Amorphous: Spatial texture, motion and words," *Organised Sound,* vol. 22, no. 3, pp. 336-344, 2017.

[14] E. Nyström, "Low-Level Topology of Spatial Texture," in *Proceedings of International Computer Music Conference*, 2015.

[15] R. Kuivila, "Events and Patterns", in *The SuperCollider Book*, S. Wilson et. al. Eds. Cambridge, MA: MIT Press, 2011.