DEVELOPMENT OF A MORPHING TOOL FOR THE "COSMOSF" SYNTHESIZER

Sinan Bökesoy
sinan@sonic-lab.com
15/6 Dogan bey sok.
Buyukada-Istanbul

SUMMARY

The purpose of this article is to explain the development of a sound morphing mechanism for the Cosmos*ƒ* advanced stochastic synthesizer[1] in its compositional workflow. A sonic morphing process defines a metaphor of exploring the sonic shape and timbral evolution on intermediate levels between two or more sonic materials. There is no robust methodology to bring exact solutions in morphing processes however they offer a rich terrain of possibilities which could be handled within various approaches to bring interesting perceptual phenomena. Themes regarding the parameter space handling of Cosmos*ƒ*, the 3D visual interface for the morphing operation, automation and innovative live control methods will be introduced here.

1. INTRODUCTION

The term <morphologie> corresponds scientifically to study of the form and structure. A short definition of sound morphing would be applying gradual transformation from one sound as the departure to another sound as the destination. The transfer of features of one sound onto another sound in order to generate cross synthesis of hybrid timbres [1][2][3]. It is a network of methodologies and organization of diverse essays for bringing heuristic results. These solutions are naturally surrounded with restrictions and artifacts regarding the analysis methodology applied for the morphing process. [4][5][6] The answer to the question of what the best and most affordable process is not clear. The purpose here is not presenting a totalized argument about morphing of sonic materials, and revealing just the constraints of the existing morphing methods but rather showing the variety that can be created within a structured matrix of divergent processes. Here, the morphing process should not be regarded as a strict mimicking process, but as a synthesis tool with specific ease to use which can even establish a shortcut to instant creativity to the composer.

1.1. Synthesis by analysis and criteria of morphing

Sound morphing can be performed by blending the parameters of the consistant structures which do generate the source and destination sounds. An important aspect of such a hybridization process is to understand that there are several possible ways of combining and cross synthesizing things [7]. Generically the definition above assumes that the matters presented as source and destination would be concrete sound samples. The morphing between audio sample materials through analysis is a top down processing approach. Through analysis we focus to obtain several feature based audio descriptive parameters which deliver the evolution of perceptually meaningful composants over time.[8] However this information lacks in projecting the physical structural facts other then statistical measurements based on the analysis of perceptual parameters.[3] STFT assumes that all the sounds are built with accumulation of sinusoidal partials, it delivers useful base to extract perceptual parameters but does not inform us about the intrinsic parameters of the structural model of the sound. Cross synthesis which create useful results practically makes use of the extracted feature space and is therefore limited within the analysis process by proposing an incomplete representation of the source and destination materials. Neverthless, the topic spectro-morphology has gained a lot of access among composers and researchers to bring a good variety of results (Figure 1). [3][5][9][15]

Figure 1. Morphology of Wishart described on *Sonic Art* pp27.

Though the sample materials are fixed mediums, the morphing process is not necessarily between equal structures, since we are not necessarily under the hood of a unified synthesis structure which has generated these sounds. Morphing process between physical models of sounds and cross synthesis by analysis works well when the destination has similar structure in comparison to the source structure of the sound with adequate feature representations [5].

[1] Cosmos*ƒ*™ is designed & programmed by Sinan Bokesoy, available at www.sonic-lab.com

To gain more success in morphing between sample sounds, a useful restriction the 'correspondence' is a looked-after quality even regarding the digital representation (recording quality). When morphing sounds, addressing the correspondence between model parameters is important. Be it an appropriate sinewave additive model or source-filter models, the morphing process becomes smoother between musical instruments that guarantees temporal and spectral correspondence.[7] More specifically, correspondence between the analysis frames of the sounds and between spectral envelope parameters representing each frame. If there is a structural correlation between source and destination, then the parameter space handling will be the most effective for applying a gradual morphing process.

The gradual transformation of the morphing process between a rigid source and destination sample sound is expected to establish linear connections and vector translations of perceptual parameters. But some application areas of the morphing process such as when interpolating the parameters of physical models [4] the morphing process does not lead to perceptually linear results. Additional techniques for warping the interpolation is needed. The gradual morphing process itself becomes a fertile medium, however as a synthesis tool the non-linear aspects are even offering other unvisited palette of possibilities.

1.2. Morphing within a synthesis model

Wishart[15] categorizes the sounds of coherent structure, but constantly changing in their present state and still keeping their global timbre space in terms of possible variations as multiplex sonic objects. The individual composants exhibit a process of change gradually in timbre space exhibiting dynamic morphology. What methods can be proposed to morph effectively in between such sonic objects rather then top-down analysis approach using the concrete samples ? One observation is that the classification of such sounds and of their morphing process is not relevant, it does not count to the typology of the sounds as the aim is to generate unheard outcome. [14].

If we can communicate directly with the synthesis model at the level of a bottom-up sonic construction process, and apply the morphing process under the hood of the synthesis model by observing the process of change of the individual composants, the result will deliver the necessary correspondence and smoothness in gradual transformations.

We can define the current state of the parameter space as a preset of sound, reminiscent to hardware synthesizers user interfaces. The parameter morphing process incorporates handling of these structural parameters with a high level interpolation mechanism accessible easily by the user, who will scan this sonic space characteristic to the sound synthesis model. The translations of these gestures are linear and recallable operations in form of parameter space transformations. The weight of each preset destination is being calculated by euclidian distance calculation. Morphing process through the handling of the intrinsic parameter space is interesting as it offers to the user accessing the intermediate states directly on the structural level not on the concrete audio sample material.

The next chapter will introduce the Cosmos*f* synthesizer (Figure 2), which does create multiplex sonic results evolving in a multiple timescaled process within a bottom-up construction mechanism, likewise a good example of implementation of a morphing tool in such category.

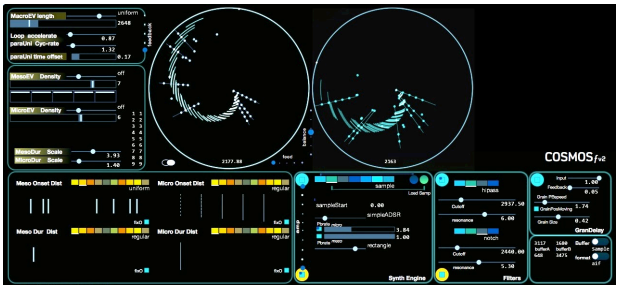

Figure 2. A part of Cosmos*f* app. Interface.

2. COSMOSF

Cosmos*f* is a multi time-scale complex event generation system (Figure 3), where the events are organized to create a sonic phenomena within controlled complexity. The event generation system on each time scale is driven by a stochastic mechanism inspired by Xenakis's formalized music tools [10] but by making benefit of modern computing power.

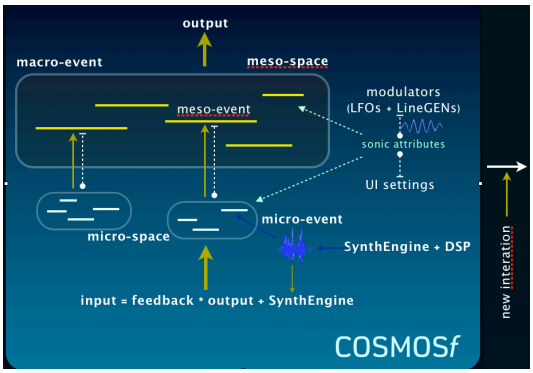

Figure 3. The basic flowchart of Cosmos*f*.

- The system creates the event space by first defining a macro space for distributing its events. Inside the macro space, meso events of certain density are distributed. (Figure 3)

These meso events also contain a micro space with certain density of micro events. This top down event distribution approach is being accompanied with a bottom-up sonic construction, where the micro events are containing the sonic data and together their micro space deliver the sonic data to each meso event.

- Various modulation sources generated by stochastic functions, LFO's and LineGEN's can be applied on micro event or meso event levels. (5 for each micro event, and 4 for each meso event). Maximum event density is 9 for meso events and 9 for micro events, which is in fact only a limitation of computer processing power. (Figure 4)

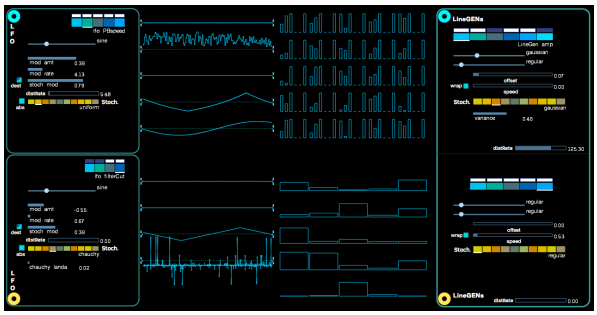

Figure 4. The multiscale modulation sources in Cosmos*ƒ*, offering complex manipulation of various parameter destinations.

- Recursive audio transfer from macro space output to the micro event entry can be applied. (Figure 5) The output buffers can capture the audio output of a macro space, freeze the captured material and deliver the content to the micro events for further processing. The length of the macro space, meso events, the micro events and their onset times can be determined with various stochastic methods offering a dynamic behavior within controlled degrees of complexity.

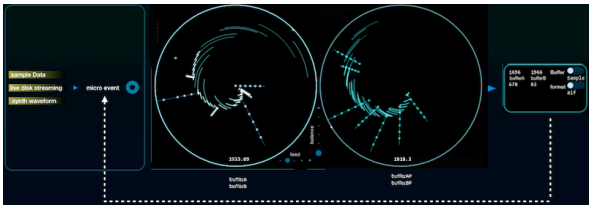

Figure 5. The output buffers of Cosmos*ƒ* deliver the macro sound back to the micro level. They can continuously feed back the signal or be frozen to deliver a static material.

- The application also lets its clock speed being altered in continuum precisely between extreme settings apart from the macro space duration and event distribution process. Time is an independent dimension in the hands of the composer for creating passages from micro to macro levels in continuous gestures.

Thanks to its coding in C++, Cosmos*ƒ* delivers particular accuracy of handling immense amount of data within a special interface design. The various facts of the model and application has been stated before in relevant resources in depth. [11][12]

3. MORPHING IN COSMOSF

Production music being created by the manipulation of concrete materials relies on organizing sonic material manually in order to create desired structures with temporal aspects and morphologies but with hand-made interaction. Cosmos*ƒ* sonic universe incorporates a dynamic morphology, which is practically hard to obtain with such methodologies. In fact, the purpose of Cosmos*ƒ* is to deliver to the composer the computing power to create generative processes, which is not possible to achieve by hand. The design of the morphing process in Cosmos*ƒ* permits the composer to focus away from low level parameter selection to higher-level control structures offered by Cosmos*ƒ* user interface.

3.1. Structural facts if Cosmos*ƒ* on the morphing

An instance of the morphing process points to a certain state between two or more sounds/preset parameters which can be represented as a multi-dimensional vector in the parameter space. Vectorization of sound as within the definition of sonic entities and their accessibility problem in the timbre space has been investigated before [9][10]. Xenakis has proposed as a unifying concept the design process for the transformation of sound. He takes instances of sound as a quantifiable and controllable entity (Figure 6) by proposing the vector definition of sonic entities with offering 4 independent dimensions e.g. the *Er(c,h,g,u)*: the timbre or instrument family, the pitch, the density and the duration depending to an axis of lexicographic time.

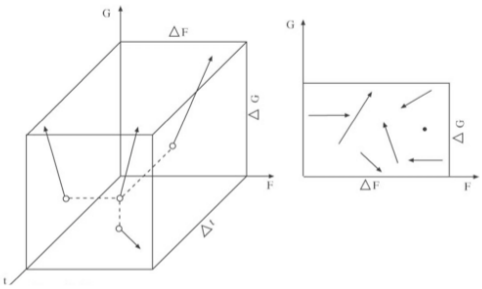

Figure 6. Grains in the timbre space distributed in a limited volume (Xenakis, Formalized Music, Fig. ii-12, pp55)

The function *F(Er,t)* defines the temporal behavior of the pointed sonic entity in this multidimensional timbre space.

3.2. Incomplete representation

The vector dimensions of the morphing pointer in Cosmos*ƒ* do include all the parameters available on the user interface. We can categorize them as parameters effecting directly the perceptual attributes, the event distribution process and signal processing/modulation parameters. According to cybernetics approach; a system subjected to complete control is necessarily incompletely represented, which is a problem of the construction of our knowledge about the system under observation [17]. Among the perceptual attributes, which Cosmos*ƒ* cannot address precisely, the pitch is a parameter where its perceptive phenomena is dependent on the temporal scale of the sonic entity carrying this information. The micro-time scale for the definition of pitch is inadequate. The organization of micro sonic entities horizontally and vertically, can create an illusion of perceptive phenomena by destructing the perception of pitch and constructing emergent perceptive properties. (integration/segregation and sequential organization) [13].

On Cosmos*ƒ*, pitch can be defined as a sample playback speed on the level of micro events and also at the level of the additive synthesis partials with defined frequency value. However on higher levels the projection of this dynamic system driving a dynamic morphology, events do break the connection with the departure pitch value of sonic entities (if perceivable) and the various local pitch values perceived on different levels. Finally pitch on the level macro is not a dimension of the morphing vector.

Another perceptual parameter, amplitude/intensity can be set on each level of event distribution. But again, like the pitch it does not exist as a macro level dimension of the morphing vector.

In a synthesis model with a bottom up construction of audio and recursive treatment of sound signals, the process of change against time becomes itself an act of the sonic design. The morphing vector interacts with the event distribution mechanism and modulation sources for the following time factors :

- *onset and duration times on micro, meso, macro level*
- *speed of micro and meso event playback* (where time change here will be expressed as pitch manipulation)
- *speed of modulation sources* : extreme values define various additional phenomena such as frequency or amplitude modulation or spectral change with secondary phenomena.

But the clock speed of the application has been kept as an independent control of the structural mechanism and therefore of the control of the morphing pointer vector. The final form of sound becomes a function of *E(clk(t), M(t))*, where *clk(t)* is the clock-rate over time and *M(t)* is the morphing vector over time.

3.3. The morphing process in recursive audio mode

The recursive sonic process implemented in Cosmos*ƒ* offers a nonlinear sonic state for the morphing pointer which points to a linearly related dimensional parameter space. A non-linear state cannot be restored by manipulating the controllable parameter space without any memory function calling to the previous sonic states. The end material is not fixed and pointing on its parameter space in a steady parameter space signifies only the departure point of the process as the initial parameters. Therefore naming this process as 'morphing' between defined destinations is a blurred definition, as the destinations are changing continuously.

4. CONTROL OF THE MORPHING MECHANISM

Cosmos*ƒ* V2.2 morphing tool introduces a spheric space where the 4 presets of Cosmos*ƒ* is being located on the surface of the sphere by representing the corners of a tetrahedron. (Figure 7) A spherical space by its polar coordinates is defined like;

x = r * sin (theta) * cos (phi)
y = r * sin (theta) * sin (phi)
z = r * cos (theta)

where the x,y,z are the cartesian coordinates retrieved from the spherical coordinates, radius *r* , inclination *theta* and azimuth *phi* with their range by definition :
r [0,1] , *theta* [0,2PI] and *phi* [0,PI].

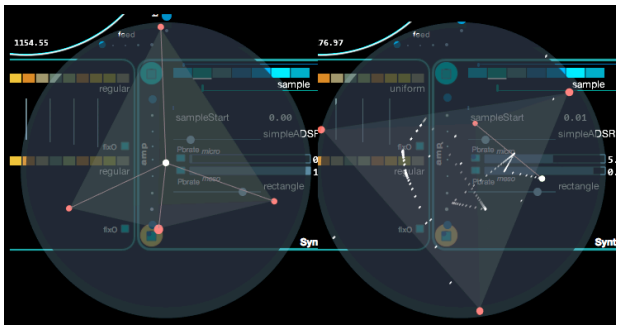

Figure 7. The 3D morphing sphere of Cosmos*ƒ*. The tetrahedron corners can be seen as red spheres representing each assigned preset. The morphing pointer is the white sphere and the lines connected to it represent the weigth of each preset. The interface shows the last 150 positions of the pointer by fading it to gray.

By changing the polar coordinate angles *phi* and *theta*, and the *radius*, one can access to every point inside this morphing sphere. The user can load different presets saved in the preset library to assign to each corner of the tetrahedron. The weight of each preset parameter destination is being calculated by an euclidian distance calculation, using the morphing pointer position and the defined tetrahedron corner coordinates.

The C++ arrays *morphSlotINT[ii][i], morphSlotFLT[ii][i]* and *morphSlotSTR[ii][i]* are holding the integer, float and string valued parameters of the 4 presets.

For instance to calculate the weighted float valued parameters according to the morphing pointer this routine will be used;

```
for (int ii = 0; ii<500; ii++) {
        morphSlotFLT[ii][4]=
morphSlotFLT[ii][presetA]*dist[0]+
morphSlotFLT[ii][presetB]*dist[1]+
morphSlotFLT[ii][presetC]*dist[2]+
morphSlotFLT[ii][presetD]*dist[3];

    }
```

where the *dist[0],dist[1],dist[2]* and *dist[3]* are the normalized distances of the pointer from the tetrahedron corners as delivering the weight for each preset parameter. Their sum is always equal to '1'.

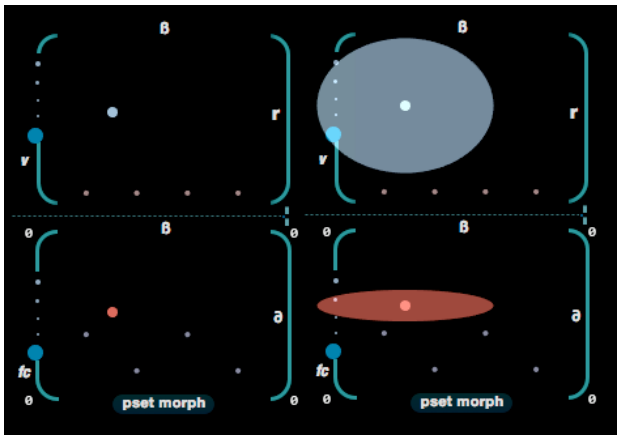

Figure 8. Adjustment of polar coordinates on Cosmos*f* user interface. The red and blue circle can be dragged by mouse precisely. The ellipse around them represent the stochastic distribution range affecting each circle. With the vertical sliders one can filter the movements of the morphing pointer and also change the speed of the stochastic distributions.

4.1. Manipulation of the morphing pointer

The morphing pointer can be moved with the mouse by adjusting the polar coordinates *phi*, *theta* and *radius* as shown on the Figure 8. Adjusting the '*phi*' or '*theta*' alone lets us move the morphing pointer on circles of longitude and lattitudes reminiscent to geographical positioning with radial distance '*r*'.

Adjusting the radial distance 'r' alone will move the pointer on a line through the center of the morphing sphere. Also Cosmos*f* offers the user some mathematical functions calculated and updated in audio-rate, which can serve as perfect paths moving the morphing pointer along with by only adjusting the speed of the movement.

All these interactions offer interesting phenomena of accessing the polar coordinates.

4.2. Stochastic morphing

The use of stochastic distribution functions on the spheric parameters lets us to move the morphing pointer randomly in respective regions of the sphere. Cosmos*f* has many probability distribution functions integrated in its mechanism such as in order to define the polar coordinates for the morphing state we can apply them directly on the parameters *phi*,*theta* and *radius*. Likewise a rich indeterministic approach will be introduced where it can simulate the probability that the sonic state points in the clouds of probability functions formed in 3D visual space of the morphing sphere (Figure 9).

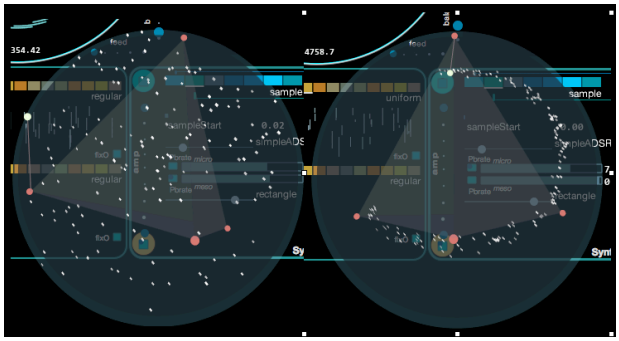

Figure 9. Random modulation of the polar coordinates. On the left all the three are uniformly modulated, on the right the '*phi'* angle modulation is dominant, and radius *r* is static.

The user defines the range of the distribution functions applied on the relevant polar coordinate. (Figure 8) It is shown as the ellipse size drawn around the polar coordinate value which itself becomes the center of the ellipse on the user interface. According these settings the stochastic distribution functions modulate the position of the morphing pointer in 1D (lines, or arcs or circles) 2D (filled circles or arcs) and 3D (cones, and regions of the spherical space) volume elements or surface fields. It becomes a continuous gesture traversing the space. (Figure 10)

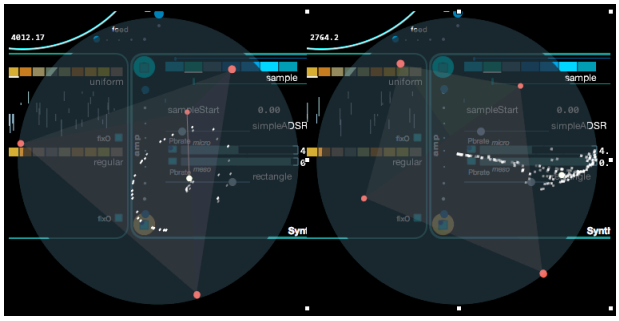

Figure 10. Random modulation can be restricted to form interesting phenomena of the moprhing pointer movement in 3D space.

The control of the morphing pointer can be achieved also by external software such the graphical event composition applications [16] via the OSC protocol and therefore additional manipulation processes can be created. However this kind of control has a limitation of being in control rate and not in audio rate resolution.

Cosmos*ƒ* breaks this limitation as it does calculate the weight of each parameter for the morphing tool in audio rate. Audio rate means, that the application does the calculations and updates inside the audio routines, *44100* times a second. Its update routine interacts with the UI and also with the OSC incoming data only in control rate. The audio-routine delivers a *phasor* with lets us to maintain it as the master clock for the application. With its integrated mathematical functions, traversing in the vector space of parameters happens with highest precision and controllable change from order to disorder thanks to its stochastic features.

Controlling the process time independently for the different levels of the morphing tool lets us manipulate the following aspects independently (Figure 8);

- *the rate of the stochastic distribution functions which modulate the position of the morphing pointer*
- *the degree of the interpolation between discrete destinations inside the morphing sphere.*
- *the rate/speed of the mathematical functions for moving the morphing pointer along their path.*

Additionally we can still control the rate of these operations independent of the Cosmos*ƒ* cycle length, hence we could slow or speed up the Cosmos*ƒ* cycle, where we could keep the speed of the motion of the morphing pointer.

Likewise Cosmos*ƒ* gains another tool to create sonic emergence and perceptual phenomena as to happen due to this kind of time warping and automation possibilities. The morphing pointer becomes a tool for dynamic stochastic synthesis as the resolution of the operation equals to the sample rate, hence drawing the waveform itself as the result.

The parameter weighting of the morphing pointer applies its value depending on the micro, meso or macro level operations of the relevant parameter. Cosmos*ƒ* is listening to all these weighted parameter calculations in audio-rate but the Cosmos*ƒ* event generation or the synthesis engine reacts to these changes in micro event time or meso event time. For instance; if the meso event density has been changed, then the change is applied on the next meso space calculation. If the macro cell length has been changed, it will take its effect at the beginning of the next cycle. However the synthesis engine and modulation sources listen to the parameters at audio-rate resolution.

4.3. Behavior of the morphing tool on the synthesis engine

There are several methods in applying the result of the morphing tool. These address the organization of the synthesis engine. (Figure 11) For instance;

- The morphing tool can apply the morph between each presets micro event sample data in two different ways:

1. By applying an amplitude balance among the waveform data assigned to the micro events can be achieved between the 4 presets.
2. A spectral balance among the waveform data assigned to the micro events can be achieved.

The latter applies a basic spectral additive balancing process between concrete waveform data without doing the perceptual analysis process. The idea is to obtain intermediary results.

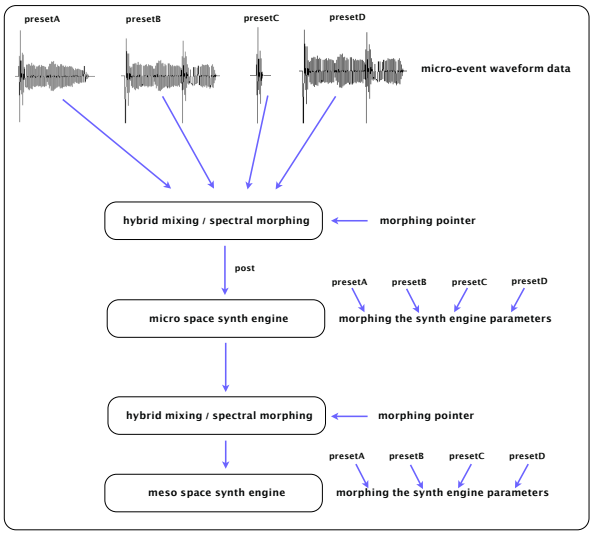

Figure 11. The morphing acting on various levels of the synthesis engine in Cosmos*ƒ*. On the parameter space and also on the waveform data by known morphing methods. The multi-scale approach is another door to emergent sonic design within this scheme.

- The amplitude balance of the 4 weighted micro event outputs can be obtained as “post” processed by the synthesis engine, and “pre” to the meso events. This is a hybrid mixing process of the 4 presets at the “pre” meso event level without balancing/morphing their micro-event waveform data. The meso event synthesis parameters, all the distribution and modulation parameters will be morphed.
- The above, plus the weighted parameter morphing of the micro event preset data. This is a hybrid mixing of the morphed presets with according to the weighted parameters among all the preset data.

In summary, the morphing tool tries to apply the morphing process on various levels on the defined input data from the user. This way, a rich palette of intermediate results can be achieved in real time.

4.4. Always there is room for improvements

The morphing tool has a potential to balance and create new parameter space among the chosen presets by combining different strategies such as hybrid feature mixing, morphing and operating on different time scales according the structural levels micro, meso or macro. The audio rate operation and modulation by stochastic functions, automation with mathematical functions and the manipulation of the clock rate, all offer unique results under a unified synthesis engine tool. Regarding the criteria of the morphing processes introduced in the introduction section, the morphing tool being implemented for Cosmos*ƒ* has still room for many improvements, and primary goals are now developing;

- *the ability of the user to choose features with different weights in order to influence Cosmosƒ behavior.*
- *specification of different distance calculation strategies for the parameter space or a selection of it.*
- *a surround sound support benefiting from the 3D morphing space.*

5. CONTROLLING THE MORPHING PROCESS WITH EXTERNAL HARDWARE

As Cosmos*ƒ* parameter space can be controlled with OSC compatible hardware, the aim here is to provide the morphing process as a performance tool, acting on the software parameters at once with global simple gestures in realtime. Since a while, 3D infrared depth sensors / time of flight cameras has arrived as end user products. With the help of such modern control tools which can offer a 3D touch free control environment, one can develop a perfect platform for the control of the Cosmos*ƒ* morphing tool which has a 3D visual interface. (Actually mouse/touch pad operation within a 3D environment is an indirect medium.)

5.1. Cosmos*ƒ* and Kinect[2] / SigmaNIL[3] library

The SigmaNIL vision framework operates with any depth sensing camera (Kinect, AsusXTion Pro Live etc.) and provides tools for finger level precision hand shape recognition, hand gesture recognition and hand skeleton tracking by analyzing the depth image data coming form the sensor device. Using the hand shape recognition module of SigmaNIL we can recognize the following shapes to start modulating the morphing parameters of Cosmos*ƒ*. Performance of these shapes is the initial step to put the overall state of the application in live editing mode. (Figure12)

Figure 12. Any shape in similarity with the above finger compositions will be recognized as the initiation of the editing mode.

While in editing mode, each distance between the finger tips, thumb-index-middle, are mapped to the assigned morphing parameter of Cosmos*ƒ*. The distance between the finger tips are being calculated by using the finger tip joints provided by the SigmaNIL hand skeleton module.

All handshapes and hand skeleton targets used for this editing process require a certain training process by using the embedded SigmaNIL tools. Different persons have applied these shapes in a repetitive session in front of the camera for a certain duration. (Figure 13)

Figure 13. Depth field image retrieved from theKinect sensor and the semantic mapping of the for the hand shape by SigmaNIL.

These depth images are the input for the training process, doing the analysis and finding out the best feature candidates defining the hand shapes based on the hand skeleton data for these specific targets. (Figure 14)

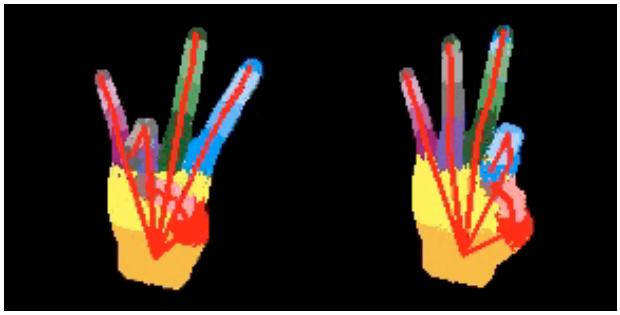

Figure 14. Hand skeleton construction and distance calculation according to the joint data by SigmaNIL.

[2] Kinect™ sensor , Microsoft Corporation 2010.
[3] SigmaNIL™ Framework by SigmaRD Computer Vision & Graphics.

To exit the editing mode with the current state of the parameters, one possible solution is that the user turns the hand by keeping the relevant finger distances around the wrist so that the thumb tip position appears higher on the 'y' axis relative to the other finger tips.

The analysis speed is limited by the camera frame rate. In order to increase the precision of the editing process, the user should hold the hand close to the camera in order to capture the hand in full sight e.g. 640X480 resolution. The precision is also depending on the distance from the camera, since the infrared beam dispersion angle is a factor of the strength and precision of the 3D scanning and does operate better in near field.

The mapping the relevant data to Cosmos*ƒ* morphing space happens via OSC transfer at the moment.

5.2. Cosmos*ƒ* and Leap Motion Controller

At the time of writing this article, we had the hands on a developer version of the Leap Motion[4] unit. It is a desktop unit, which offers the tracking of all the finger position data along with the orientation and 3D positioning on the air. The SDK of Leap Motion allows to implement the captured high-level streaming data in a C++ environment easily.

Via the Leap Motion Controller, the morphing pointer can be controlled by moving the tip of a finger in 3D space. The finger tip position is being mapped to Cosmos*ƒ* spherical morphing environment and converted to polar coordinates. (Figure 12)

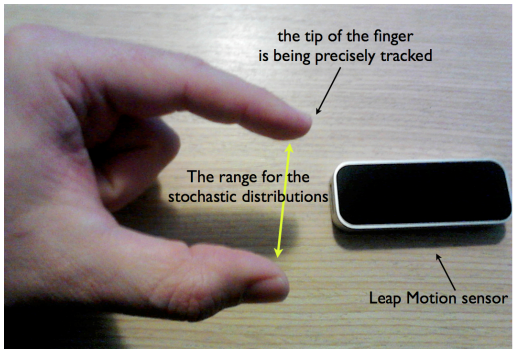

Figure 15. The Leap Motion unit and an example gesture controlling the range of distribution functions modulating a polar coordinate of the morphing tool.

The application can track the number of hands and fingers on the scene. The following gesture controls have been implemented.

- *Single Hand - 1 Finger pointing* : Morping Pointer position data mapping in 3D space.
- *Single Hand - 2 Finger range definition* : defines one polar angle stochastic distribution range
- *Double Hand - 2 Finger range definition* : defines polar angles stochastic distribution range for each hand
- *Double Hand - 2 Finger radius definition* : The distance between hands define the radius stochastic distribution range.

This simple and short code is an example of extraction of the positioning data and processing it.

```
if( leap.isFrameNew() && simpleHands.size() ){

if (simpleHands.size() == 2){
for(int i = 0; i < simpleHands.size(); i++){
if(simpleHands[i].fingers.size() == 2){

for(int j = 0; j < 2; j++){
int id = simpleHands[i].fingers[j].id;
ofVec3f pt = simpleHands[i].fingers[j].pos;
            pp[i][j] = pt;
            hand1distance = pp[0][0]-pp[0][1];
            hand2distance = pp[1][0]-pp[1][1];
            hand1med = (pp[0][0]+pp[0][1])*.5;
            hand2med = (pp[1][0]+pp[1][1])*.5;
            }}}
```

Finger 1 and *Finger 2* vectors are substracted and the length of the resulting vector is calculated on each hand, giving the range data for the relevant gesture. Also the middle space point between *Finger1* and *Finger2* is assigned as a position data to calculate the distance between two hands.

These are the most simple implementation schemes. Many further gesture analysis and teaching methods can be applied for controlling additional parameters of Cosmos*ƒ* based on the knowledge of previous research on the field. As a first impression, the control in a 3D touch free environment works robust and precise due to the impressive frame rate of the data delivered from Leap Motion. It encourages us to continue the development of the 3D user interface.

6. REFERENCES

[1] M. Caetano and X. Rodet, "Automatic timbral morphing of musical instrument sounds by high-level descriptors", in Proc. ICMC, 2010.

[2] ——, "Sound morphing by feature interpolation", in Proc. ICASSP, 2011.

[4] Leap Motion Inc. 2012, www.leapmotion.com

[3] J. M. Grey and J. W. Gordon, “Perceptual effects of spectral modifications on musical timbres”, Journal Acoust. Soc. Am., vol. 63, no. 5, pp. 1493–1500, 1978.

[4] T. Hikichi and N. Osaka, “Sound timbre interpolation based on physical modelling”, Acoustics Sci Technology, vol. 22, no. 2, pp. 101–111, 2001.

[5] N. Osaka, “Timbre interpolation of sounds using a sinusoidal model”, in Proc. ICMC, 1995.

[6] E. Tellman, L. Haken, and B. Holloway, “Morphing between timbres with different numbers of features” Journal Audio Engineering. Society, vol. 43, no. 9, pp. 678– 689, 1995.

[7] M. Caetano, N. Osaka, “A Formal Evaluation Framework for Sound Morphing”, in Proc. ICMC, 2012.

[8] Bresson, J., *Representation et Manipulation de Donnees d’Analyse sonore pour la Composition Musicale* ; *the perceptual organization of sound*. Rapport de stage IRCAM – Ecole Superieure en Sciences Informatique – Universite de Nice – Sopphia Antipolis, 2003.

[9] Grey, J. M. « Multidimensional Perceptual Scaling of Musical Timbre » in: *Journal of Acoustical Society America*, 1977.

[10] Xenakis,I. *Formalized Music.* Pendragon Press, Hillsdale, NY, 1992.

[11] Bokesoy, S. "Feedback Implementation within a Complex Event Generation System for Synthesizing Sonic Structures", *Proc. of Digital Audio Effects (DAFX’06)*, Montreal, Canada,pp. 199-203., 2006.

[12] Bokesoy, S. « Presenting CosmosF as a Case Study of Audio Application Deign in Openframeworks» in : *Proceedings of the International Computer Music Conference ICMC’12*, Lubiana, pp. 197-203, 2012.

[13] Bregman**,** Albert. *Auditory Scene Analysis* ; *the perceptual organization of sound*., Cambrigde, MA, USA : MIT Press, 1990.

[14] Schaeffer, Pierre. *Traité des objects musicaux*. Paris: Seuil, 1977.

[15] Wishart, Trevor. *On Sonic Art,* ISBN 371865461, 1987.

[16] Coduys, T., and Ferry, G. ‘IanniX aesthetical/symbolic visualisations for hypermedia composition', Proc. International Conference Sound and Music Computing. Paris, 2004.

[17] Wiener, Norbert. *Cybernetics (Control and Communication in the Animal and the Machine),* MIT Press, Cambridge MA, 1948,1961.