

VISUALIZATION OF SOUND AS A CONTROL INTERFACE

Anne Sedes, Benoît Courribet, Jean-Baptiste Thiébaud

Centre de recherche Informatique et Création Musicale
Université de Paris VIII

Maison des Sciences de l'Homme Paris Nord, Saint-Denis, France
{asedes | bcourribet | jbthiebaut}@mshparisnord.org

ABSTRACT

We here introduce the opportunity of using visualization of sound as a control interface, for artistic live performance as well as for new digital audio effects developments. Two different approaches are exposed. The first access consists in using video matrixes for mapping the variables parameters of sound processing, with the coordinates of any controller in a 2D plane or 3D space. The second access proposes a visualization of sound that modifies sound data by processing the data of the image itself with its own graphical properties.

Exploring this kind of « transducing » relation between visual and audio may be interesting for artistic creation domain using virtual surroundings; it may cause an interest for the real time digital audio, for audiovisual mixing and new interfaces for sound design. Besides, it points the opportunity of developing 3D control interfaces for audio and visual processes.

1. INTRODUCTION

Beyond scientific graphical representations of sound, such as waveforms or spectrograms, there is no objective representation of sound phenomenon, no machine to visualize sound. There are only subjective, more or less metaphorical representations, in heterogeneous sound or musical context. Visualizing the sound phenomenon, after the real time digital audio flow coupled to real time video data flow, the way a programming software such as Max/MSP/Jitter [1] can offer, consists as far as a representation, to visually mark some of the sound qualities, to offer a better listening, a better perception, and real time operation on sound. Hence, we are inverting the ordinary relation between the audio and visual domains, by using visualization of sound as a control interface of sound perceptive parameters, indeed exploiting the graphical properties of sound visualization to process sound itself.

2. USING TERRAINS AS VISUAL INTERFACES

Considering our previous research on self-centered perception of sound spaces [2][3][4][5], we have decided to link some of its characteristics with our study of visualization of sound as an interface. Assuming that a musical piece can be seen as a set of parameters varying in time, a deterministic musical piece can be represented as in Figure 1. There, the time line is explicit, and proposing this type of visualization modifies our perception of the structure of the piece (because all evolutions past and future can be seen at the same time). Thus, we have decided to let the user free to imprint his own timeline to the piece. Referring conceptually

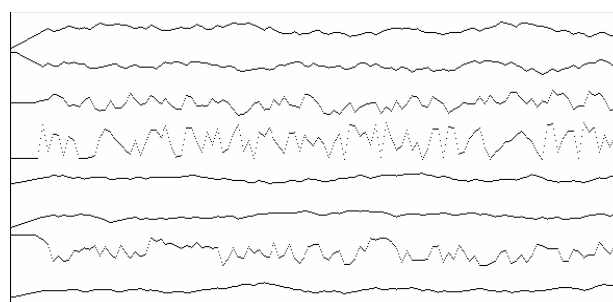


Figure 1: A time/amplitude representation of the evolution of 8 parameters.



Figure 2: A software interface for terrain design.

ally to the Wave Terrain Synthesis [6], we have decided that each parameter would be represented by a terrain. The terrain will appear on screen as a monochrome picture, the value of the parameter being mapped with the intensity of the colour. A terrain set (a superposition of terrains) would result, on which the user could move, outputting the parameters values at the point where he is located.

Thanks to the Max/MSP/Jitter programming environment, we have developed software to illustrate these concepts, using the OpenGL set of Jitter objects. We are now going to describe successively both the writing and the visualization interface.

The main goal of the writing interface is to create quickly a terrain set by drawing coloured shapes within a reduced matrix (a tenth of the final matrix).

One has to select one of the three planes (red, green and blue) and set the intensity of the corresponding colour and then draw directly onto the terrain window (the left pane in Figure 2). It is possible to use an interpolating filter in order to smooth the terrain

for the final matrix (the result of the interpolation appears in the right window in Figure 2). Thumbnails for each plane are available on the left side. When done, the resulting matrix has to be exported in the Jitter matrix format in order to be loaded later in the visualization interface.

The visualization interface is a minimal 3D environment consisting in a textured ground on which it is possible to move thanks to mouse. The 3D engine is the same as the one used in prior works *Egosound* [7] and *La Terre ne se meurt pas* [8].

For an (x,y) set of cartesian coordinates corresponding to the location of the user on the ground, we have a set of parameters $R(x,y)$, $G(x,y)$ and $B(x,y)$ according to the matrix (x,y) cell values. From now, we will only consider one parameter $P(x,y)$, because of the grey scale constraint of the presentation. In Figure 3 we can see a terrain (one parameter) together with its projection in the visualization interface.

From a trajectory $T: t \rightarrow (x(t),y(t))$ (a set of (x,y) coordinates varying in time), a time-varying $P(t)$ will result. We just have seen in a very simple way how time and space are linked, and how we can recover a time-varying expression of a parameter from its spatial distribution. The screenshots in Figure 4 show the same terrain, on which are drawn two different trajectories (clockwise). Thus, we have two different evolutions of the parameter, represented first with colour intensity, then as usual amplitude / time graphs. Therefore, we can easily imagine that, given a terrain there is an infinity of possible dynamic evolutions for its associated parameter. Of course, given a fixed trajectory, depending on the variation of the speed along this trajectory, many different results can be obtained (see, e.g., Figure 5).

Moreover, this has been discussed in the case of a single parame-



Figure 3: A terrain seen "from above" (left) and this terrain used as a texture in the visualization interface (right).

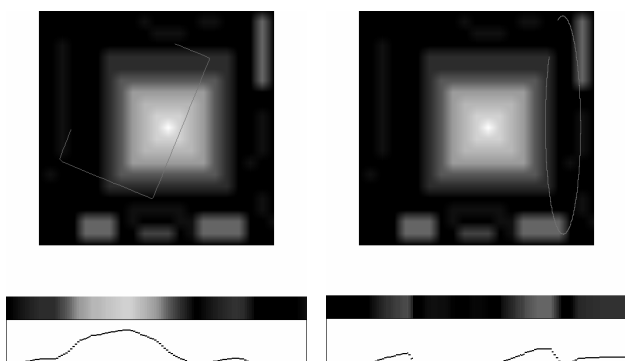


Figure 4: Two different trajectories on the same terrain and the resulting amplitude/time representations of the parameters evolution.

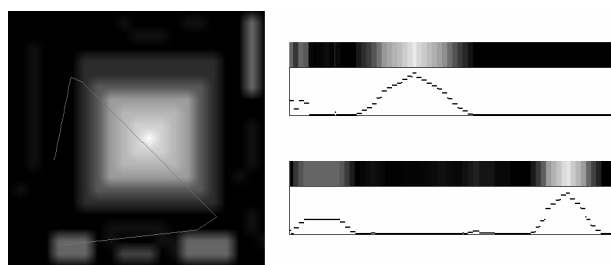


Figure 5: The same trajectory (clockwise) with different speed variation and the associated evolution of parameters.

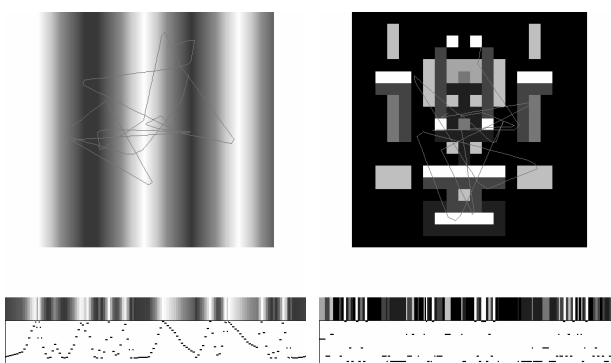


Figure 6: Two different terrain design strategies.

ter, and can be extended to the case of sets of many parameters. Although there might be an infinity of musical results associated with a single terrain set, it doesn't mean that the piece (even if interactive) escapes to the composer intentions. On the contrary, we will show briefly how the design of a terrain can determine global trends. The only thing known for sure is that the movement on the terrain is continuous: the (x,y) pairs will evolve contiguously (the user cannot "jump" from one point to another). Here are two examples of strategies for terrain design. In the left screenshot of Figure 6 the terrain has been designed with care of continuity, therefore, whatever trajectory or speed will be used onto it, the associated result will be a continuously evolving parameter, while in the right screenshot no interpolation has been used and small shapes have been drawn; the parameter will then evolve with discontinuities. Thus, one can determine trends in a certain way and leaving room for variations.

One of the main interests of seeing parameters as terrains is that we can use many different software packages to create 2D pictures to be considered as terrains. The one we have presented has been developed for an almost didactic purpose and has the advantage of being integrated in the Jitter environment. Of course, we could use software such as Photoshop [9], Scilab [10] or terrain editors like the Parametric Terrain Editor (PTE) [11] which has been especially developed for terrain design in 3D software. An example of terrain designed with PTE is shown in Figure 7.

Though the visualization interface is based on an immersive 3D environment, we have only dealt with 2D geometry. It is possible to imagine that the value of a parameter might be used to set both the colour intensity and the z coordinate of the corresponding point on the terrain (see Figure 8). However, we are working on the realization of tools for "real" 3D design, which implies many

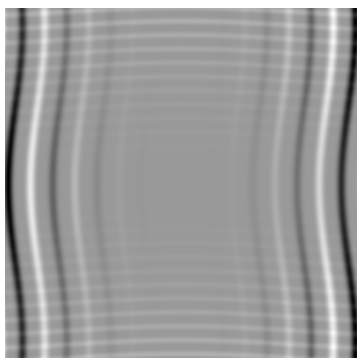


Figure 7: Example of a terrain designed with the Parametric Terrain Editor.

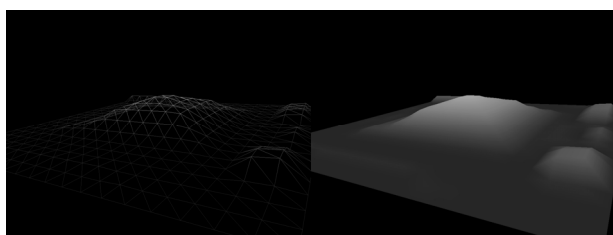


Figure 8: Two views of a 3D terrain.

conceptual problems (though the visualization interface is ready by now for 3D movements). We can also imagine time-varying terrains (videos instead of still images) setting a global timeline for the interactive piece, so that sequences might not be experienced twice, for example.

3. GRAPHICAL TRANSFORMATION OF AN ANALYSED SOUND

We have presented visual interfaces that allow the user to control high-level sound parameters. We will now present a visual interface based on a formal representation of sound. Our aim is to modify the sound helped by a graphical interface of the sound

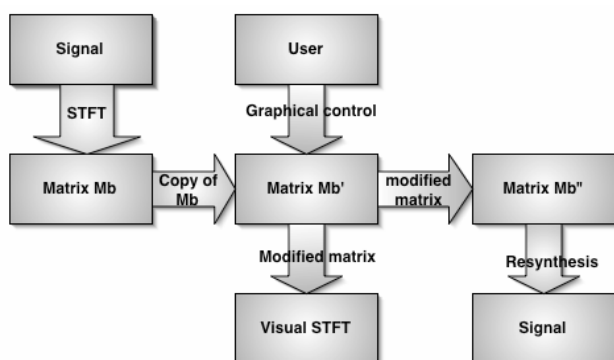


Figure 9: The user controls the matrix with graphical operations. Transformations are applied on both image and sound.

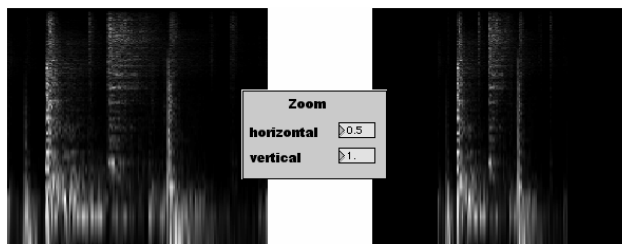


Figure 10: The right image is a zoom out of the left image: the sound is time-reduced.

itself after analysis. In order to manipulate the sound graphically, we apply a STFT to a stream of input samples and draw it on a visual window. Scaled to a logarithmic view, the sonogram is then graphically modifiable. The resulting matrix of this transformation is then synthesized (see Figure 9).

The window representing the sound is a time-frequency-amplitude representation. Graphical transformation would necessarily transform sound according to these three parameters. Considering the visual window as a drawing interface, the user modifies the STFT synthesized in real-time. We developed mainly two ways of manipulating the visual STFT.

3.1. Global transformations

We present the different modifications according to the main transformation they concern between time, frequency and amplitude. Some similar operations have been described [8]. They are usually obtained by a phase vocoder, with functions that modify partials. Regarding sound as an image allows us to apply graphical transformations that have not been implemented within a phase vocoder. The interface is transparent due to the formal relation between the image and the sound. Graphical transformations are obtained with standard externals of the Jitter library. Based on Musical transformations using additive analysis / resynthesis of the Computer Music Tutorial [6], we describe the transformations obtained with a graphical control.

3.1.1. Time-based transformations

Considering that the abscissa axis represents the time, all the graphical modifications along this axis will modify the sound.

Time scaling (see Figure 10): Zoom in or zoom out. A zoom in causes a graphical interpolation that sounds like an ordinary deceleration. A zoom out causes a graphical reduction that sounds like acceleration. Due to the time limit of the matrix, in the zoom transformation loss of information occurs. While zooming in, output parts of the matrix are lost. While zooming out, the image is reduced, thus data are lost too.

Non-linear synthesis (see Figure 11): The user can control the time position of the STFT synthesis. The user controls the index of a vector of the matrix, corresponding to a single vertical line of the image. The synthesis thus concerns only 512 samples (0.011"). The time line can be modified by a control of the index. For example, a broken time line can be used to control the STFT synthesis. In the particular case of the synthesis of a single line, the repetition of 512 samples is a very textured, continuous sound.

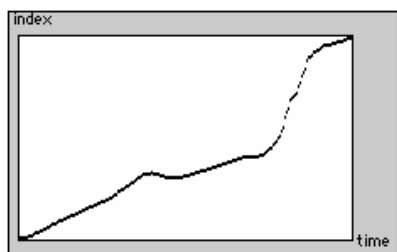


Figure 11: The ordinate axis represents the index of the synthesized vector. A non-linear curve causes a non-linear time-play of the sound.

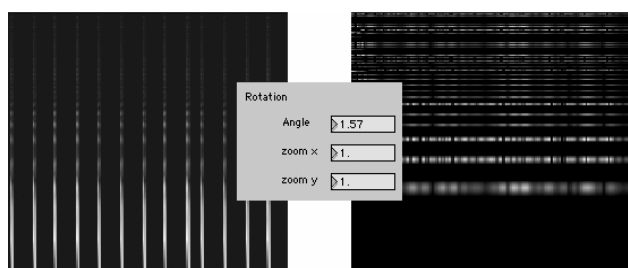


Figure 12: The left image is the STFT of a digital click. Transient attack becomes continuous sound on the right image after a $\pi/2$ rotation.

3.1.2. Frequency-based transformations

Considering that the ordinate axis represents the frequency, all the graphical modifications of this axis will modify the sound.

Spectrum shifting: Shifting the image up or down. The modified sound consists of the addition of a factor n to all partials. This shifting can sound differently according to the algorithm used for scaling the STFT.

Spectrum inversion: Vertical inversion of the image. Obtained by a vertical zoom of a factor -1 (low frequencies take place of high frequencies, and vice versa)

Spectrum zoom: Zoom in or zoom out of the ordinate axis. In a zoom in operation, the frequency domain zoomed is scaled to a larger frequency domain. Border information and precision are lost due to the finite definition of the STFT. In a zoom out operation, the frequency domain is restricted. Data are scaled to a lower domain.

Spectrum rotation (see Figure 12): Rotation of the image based on an anchor point. It causes a linear frequency variation, ascendant or descendant according to the sense of rotation. For a $\pi/2$ rotation, a transient attack becomes a continuous sound, and vice versa.

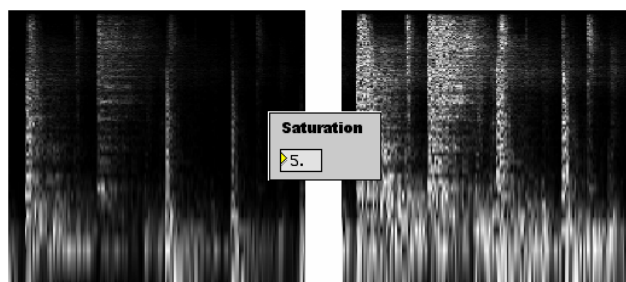


Figure 13: The saturation factor enhances the luminance of the image. For small values, the result (on the left) sounds much like amplification.

3.1.3. Amplitude-based transformations

Considering that amplitude is a function of the luminance of the image, all the graphical transformation on luminance would influence the sound.

Kind of reverb: Apply a blur transformation to the matrix.

Saturation (see Figure 13): Saturation / desaturation of light. The matrix is multiplied by a factor n .

The limits of this approach are that this sonic design can be reduced to the drawing of an FFT. Creating sounds with this process encounters graphical needs imposed by the time-frequency representation of the FFT.

3.2. Local transformations

In order to apply local transformations, we use a transfer matrix that allows the user to draw graphic controls, as shown in Figure 14. The transfer matrix uses the R, G, B layers as different planes of parameters. While the user draws on the transfer matrix, the STFT is processed with each plane of parameters, according to different ways that we will develop now.

We consider that the abscissa axis of the transfer matrix represents time the same way it is represented in the STFT. All the transformations we draw on this interface will affect the sound in time.

The first transformation we program concerns amplitude. We

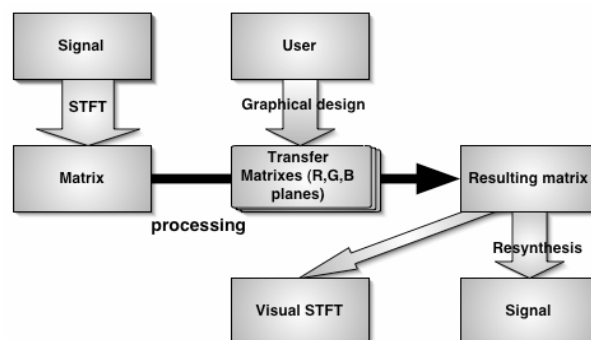


Figure 14: The three planes of the designed image processing of the basic matrix.

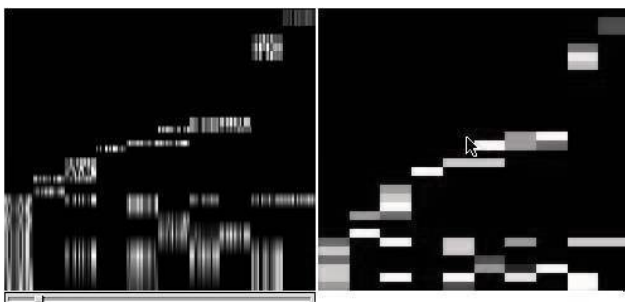


Figure 15: The right image is the interface of the transfer matrix. The left image is the result of the basic STFT multiply by the transfer matrix.

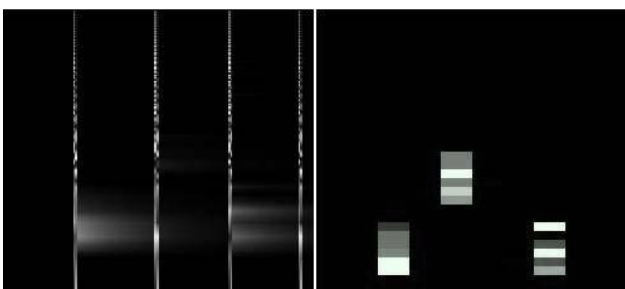


Figure 16: Green coloured cell (right image) cause a delay on the STFT (left image).

use the red plane of the matrix transfer as a frequency filter. The value of each cell, between 0 and 255, is scaled to a multiplying k factor between 0 and 1. The STFT matrix is then multiplied by the red matrix to obtain an amplitude-filtered matrix that can be synthesized (see Figure 15).

The green plane is used for a spectral delay. Each cell of the matrix, also scaled to the interval $[0,1]$, will process the STFT according to a jitter external, `jit.scanslide`. The STFT bins will be delayed graphically according to the green bins of the transfer matrix. This causes a spectral delayed signal (see Figure 16).

These transformations have been implemented in the Sonos software. The blue plane has not yet been devoted to a specific effect. This approach of sound transforming by a mapping of graphic properties to a musical purpose shows a great potential. Image processing used by video-jokeys could control musical parameters via this interface.

4. REFERENCES

- [1] www.cycling74.com
- [2] A. Sedes, V. Verfaillie, B. Courribet, "Visualisation de l'espace sonore, vers la notion de transduction," in *Espaces sonores, actes de recherches*, sous la direction d'Anne Sedes, éditions musicales transatlantiques, pp. 125–143, Paris, 2003.
- [3] A. Sedes, B. Courribet, J.-B. Thiébaut, "Visualisation du sonore avec le logiciel egosound: work in progress," in *Actes*

des journées d'informatique musicale 2003, Montbéliard, 2003.

- [4] A. Sedes, B. Courribet, J.-B. Thiébaut: "Visualisation sonore avec Egosound, entre recherche et creation," in *Hypertextes, hypermédiat, créer du sens à l'ère numérique*, sous la direction de J. P. Balpe et I. Sahlé, Hermès/Lavoisier, pp. 247–274, Paris, 2003.
- [5] A. Sedes, B. Courribet, J.-B. Thiébaut, "Egosound, an ego-centric interactive and realtime approach of sound space," in *Proceedings of the 6th International Conference of Digital Audio Effects (DAFX-03)*, London, 2003.
- [6] C. Roads, *Computer Music Tutorial*, MIT Press, Cambridge, Massachusetts, 1996.
- [7] Egosound is a software downloadable from www.mshparisnord.org
- [8] *La terre ne se meut pas* is an artistic virtual environment, for the exhibition "Créer du sens à l'ère numérique", *H2PTM'03*, Saint-Denis, France.
- [9] www.adobe.com
- [10] www.scilab.org
- [11] www.castironflamingo.com/tutorial/pte