Interaction-optimized Sound Database Representation

Ianis Lallemand, Diemo Schwarz

INTERACTION-OPTIMIZED SOUND DATABASE REPRESENTATION

Ianis Lallemand,
UMR STMS
IRCAM–CNRS–UPMC
Paris, France
ianis.lallemand@ircam.fr

Diemo Schwarz
UMR STMS
IRCAM–CNRS–UPMC
Paris, France
diemo.schwarz@ircam.fr

ABSTRACT

Interactive navigation within geometric, feature-based database representations allows expressive musical performances and installations. Once mapped to the feature space, the user's position in a physical interaction setup (e.g. a multitouch tablet) can be used to select elements or trigger audio events. Hence physical displacements are directly connected to the evolution of sonic characteristics — a property we call *analytic sound–control correspondence*. However, automatically computed representations have a complex geometry which is unlikely to fit the interaction setup optimally. After a review of related work, we present a physical model-based algorithm that redistributes the representation within a user-defined region according to a user-defined density. The algorithm is designed to preserve the analytic sound-control correspondence property as much as possible, and uses a physical analogy between the triangulated database representation and a truss structure. After preliminary pre-uniformisation steps, internal repulsive forces help to spread points across the whole region until a target density is reached. We measure the algorithm performance relative to its ability to produce representations corresponding to user-specified features and to preserve analytic sound–control correspondence during a standard density-uniformisation task. Quantitative measures and visual evaluation outline the excellent performances of the algorithm, as well as the interest of the pre-uniformisation steps.

1. INTRODUCTION

1.1. Background

In this study, we focus on interactive musical performances and installations based on navigation within a sound database. The user selects database elements and triggers events (for instance the playback of a sample) by physical navigation in an interaction setup, which is mapped to a geometric database representation.

Sound databases may include data as diverse as full-length recordings, samples, sound grains (used for instance by corpus-based concatenative synthesis methods [1]) or even non-audio elements such as synthesizer presets. In most cases, it is possible to build feature-based data representations by automatic and quantitative measuring of each database element's sonic characteristics. They can be quantified using sound descriptors for instance [2], or the parameters of a synthesizer preset. Such representations are closely related to the field of content-based music information retrieval, which has attracted much attention in the past years as it allows greater insight on the data than usual keywords classification.

Besides simplifying the process of working with large databases, feature-based representations have one important property: they guaranty that the elements' positions in the database representation are connected to their sonic characteristics. To illustrate this property, we consider the example of the CataRT software [3], a corpus-based concatenative synthesizer which provides screen-displayed, 2D representations of sound grains databases, obtained by computing their sound descriptor values (cf. figure 1). In this example we have used Spectral Centroid[1] and Periodicity[2] descriptors. We use the computer mouse as an interaction setup: a simple way to control the synthesis is to trigger the playback of a grain when the point representing it on the screen is hovered by the mouse cursor. Because our representation is feature-based, small mouse movements result in playing similar-sounding samples, while larger movements allow the user to pick grains with greater sonic differences. Furthermore, a gesture along a coordinate axis results in keeping constant one characteristic of the selected grains, while controlling the remaining property (for instance playing grains of same brightness but of different harmonic characters). We use the name *analytic sound–control correspondence* to define this property.

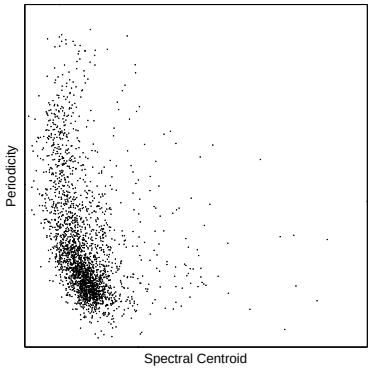

Figure 1: *Database representation using Spectral Centroid and Periodicity descriptor values.*

Common descriptions of sound data use a large number of features, which yield high-dimensional database representations. Such representations are usually not suited for physical navigation, which is usually performed in two or three dimensions. Hence new

[1]The Spectral Centroid (measured in Hertz) is defined as the center of gravity of the FFT magnitude spectrum. It is closely related to the perception of brightness.

[2]The Periodicity measures the harmonic character against the noisy character of the sound.

database representations have to be used for this purpose; they can either be obtained by dimensionality reduction methods or by direct selection of the features the user wants to control.

Our study adresses a problem relative to the use of a 2D interaction setup, though it can virtually be extended to any number of dimensions. Exemples of such interaction setups are XY controllers, multitouch tablets, or even the floor of an exhibition room combined with a position tracking device (in this case, selection would be performed by the user's displacements in the room).

1.2. Problem

Each interaction setup has its own fixed geometry: it would be a rectangle for a XY controller, a disk for the Reactable [4], or any possible shape for an exhibition room. It is very unlikely that the mapping between this geometry and the geometry of the 2D database representation obtained by dimensionality reduction or feature selection will be optimal (in a user-defined sense). Hence we need a flexible way to adapt the representation geometry to that of our interaction setup, while preserving as much as possible the analytic sound–control correspondence property — one of the strengths of feature-based representations.

To illustrate this problem, consider that we want to sonify an exhibition space using our previous example database and a 2D representation. Our interaction setup is the floor of the exhibition space. A camera is used to perform position tracking of the visitors, so that each visitor's physical navigation in the room create a path through the database representation. A sample is triggered each time this path intersects a sample's position. It is as if we had mapped the database representation to the surface of the room, and a sample was triggered each time a visitor's position intersected a sample's position. Suppose that we seek full sonification of the exhibition space, i.e. that we care to use all the interaction space that is available to us for controlling the samples, and that we want higher sample density near the top-right corner of the room. Figure 2 shows two sonifications achieved with different representations, which are superposed to the room blueprint.

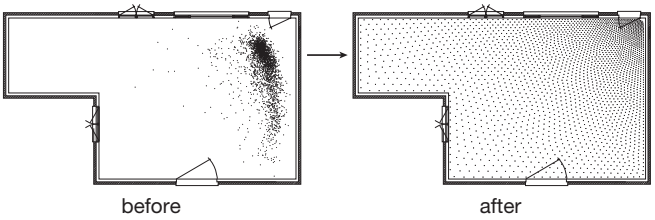

Figure 2: *Examples of sonification of an exhibition space using a 2D database representation.*

On the left, the representation shown in figure 1 has been used. Simple transformations such as symmetries and scaling obviously preserve analytic sound–control correspondence. Though we have been able in this case to put a high density region near the top-right corner, we obviously cannot obtain by these means a new representation that fits perfectly the boundaries of the room. In all cases a large amount of "silent" zones will be left, in which no samples can be triggered. Clearly, manually moving points so that we obtain a representation that roughly fits the room geometry when superposed to its blueprint is not a good solution to our problem, as it would be very difficult for us to ensure that the new representation retains some aspects of the analytic sound–control correspondence property. Furthermore, moving points manually gets more and more difficult as the database size increases.

On the contrary, a geometric algorithm might take care of automatically finding a new representation that suits our geometric constraints, while preserving analytic sound–control correspondence as much as possible. The right part of figure 2 shows a database representation obtained with *unispring*, an algorithmic solution we have provided to this problem (detailed in section 3). Besides having been able to make the new representation fit the geometry of the room, we have kept a higher density region near the top-right corner, as required by the sonification design we had in mind.

1.3. Algorithmic Solution

We present in section 3 *unispring*, a physical model-based algorithm which allows to spread the original representation points within a user-defined region. The expected data point density can be specified using the mathematical framework detailed in section 3.1.

The algorithm was designed to preserve analytic sound–control correspondence as much as possible; an evaluation of its performance is presented in section 4.

2. RELATED WORK

2.1. Previous Work

We had previously addressed the problem of locally controlling the density of a database representation using mass spring damper-based algorithms. They helped avoiding overlapping points by pushing them appart, and have been used in a dimensionality-reduction algorithm [5]. Other analytic sound–control correspondence preserving algorithms were also developed [6], but proved to be less efficient than the solution detailed in this article.

2.2. Data Visualisation Methods

Dynamic visualisation methods, such as zoom and pan, allow accurate selection of individual points in high-density regions. These approaches might also adjust the number of points to display accordingly to the zoom level, in order to avoid data overlap. However, such methods apply when the user interacts with the help of a visual feedback of the database current state, which is not always the case (as shown for instance in section 1.2).

On a more fundamental level, also note that our aim was not to provide dynamic data display, but geometrically transforming the representation of the data to obtain an optimal, "static" interface for navigation. By static, we mean that this interface is determined once by the user, and doesn't require further adjustments in the course of the interaction process.

2.3. Dimensionality Reduction

Dimensionality reduction algorithms such as PCA [7], weighted PCA [8] or MDS [5] can be used to obtain 2D representations from original high-dimensional feature-based representations. Another approach is to project the representation onto a plane by selecting the two features users will be able to control. Dimensionality reduction methods might be helpful in cases where it is unclear

which features to control, or when it appears that no pair of features amongst those immediately available by data analysis will provide satisfactory results in terms of expressivity. The latter case might particularly be true for applications using sounds of very different natures, such as those contained in a collection of field recordings for instance. However, note that projection by selection of two features naturally maps their perceptual meaning to the two degrees of freedom the user interacts with. By taking care of preserving the analytic sound–control correspondence property, our algorithm also preserves this perceptual meaning.

2.4. The *distmesh* Algorithm

The *distmesh* algorithm, available as a Matlab toolbox [9], generates unstructured triangular meshes using a physical algorithm. It is based on a simple mechanical analogy between a triangular mesh and a 2D truss structure, or equivalently a structure of springs. It provides a mathematical framework that allows the user to specify the internal geometry of the mesh as well as the region over which it has to be generated. Whereas *distmesh* is aimed at generating a mesh over a blank region, the physical algorithm part of *unispring* (detailed in section 3.3) adapts the physical model used in *distmesh* to relocate previously existing points (the initial feature-based database representation).

3. THE UNISPRING ALGORITHM

The *unispring* algorithm works in two parts. It starts by performing a pre-uniformisation of the 2D initial database representation (3.2), before iteratively applying a physical model-based algorithm (3.3). The algorithm spreads the representation points within a user-defined region, inside which the local density can be specified.

3.1. User-specified Features

The user-defined region is represented by its *signed distance function*, which gives the distance from any point in the plane to the closest region boundary. This function takes negative values inside the region and equals zero on its boundary. Analytic computation of the signed distance function is possible for simple regions, such as square and circular ones. For more complex regions, the function has to be computed numerically. We provided support for polygonal regions, using an iterative method implemented in [10].

The user can specify the final data point density by providing a desired length function $h(x, y)$. If (x, y) are the coordinates of the middle of two points, $h(x, y)$ gives a target distance between these points that should be reached after applying the algorithm. The resulting distances are in fact proportional to those specified by $h(x, y)$, which actually gives the relative distance distribution over the region. Density-uniformisation can thus be obtained by providing any uniform length function.

3.2. Pre-uniformisation

The pre-uniformisation steps provide equally spaced coordinate values on each axis. Since the expected final density of data points is user-specified, the pre-uniformisation steps can be seen as a way to provide a "neutral" distribution of data points that will allow more efficient action of the subsequent physical algorithm.

- **step 1: on each coordinate axis,**
  - **step 1a:** sort the coordinate value list $(x_i)_{1\leq i\leq N}$ (resp. $(y_i)_{1\leq i\leq N}$). For each position i, get the position $n(i)$ in the sorted list.
  - **step 1b:** fill output coordinate value list $(x'_i)_{1\leq i\leq N}$ such as $x'_i = n(i)$ (resp. $(y'_i)_{1\leq i\leq N}$).
- **step 2:** normalize the resulting coordinate values so that all data points lie inside the user-specified region.

Figure 3 shows the intermediate database representation obtained after applying steps 1 and 2 to the representation shown in figure 1.

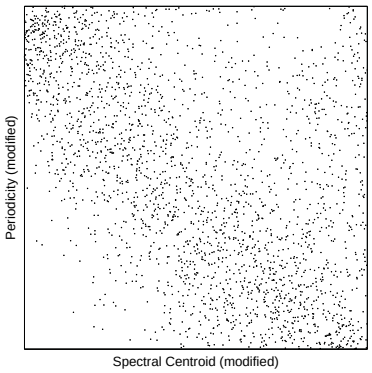

Figure 3: *Database representation using pre-uniformised Spectral Centroid and Periodicity descriptor values.*

3.3. Physical Algorithm

The *unispring* algorithm takes as input the coordinate lists $(x'_i)_{1\leq i\leq N}$ and $(y'_i)_{1\leq i\leq N}$ obtained by pre-uniformisation, and outputs new coordinate lists $(x''_i)_{1\leq i\leq N}$ and $(y''_i)_{1\leq i\leq N}$. It works according to a physical analogy: a Delaunay triangulation of the data point distribution is performed, which defines a truss structure where edges of the triangles (the connections between pairs of points) correspond to bars, and points correspond to joints of the truss.

Each bar of the truss structure has a force-displacement relationship $f(l, l_0)$ depending on its current length l and its unextended length l_0 (which is computed using the desired length function). To help points spread out across the whole user-defined region, only repulsive forces are allowed:

$$\begin{cases} f(l, l_0) = k(l_0 - l) & \text{if } l < l_0 \\ f(l, l_0) = 0 & \text{if } l \geq l_0 \end{cases} \quad (1)$$

Points that move outside the user-defined region during the algorithm iteration are moved back to the closest boundary point. This corresponds to the physical image of external reaction forces on the truss structure, that act normal to the boundary; hence points can move along the boundary, but not go outside.

The 2D truss structure obtained by triangulating the distribution bounds each point to its initial closest neighbors. Repulsive forces computed along the structure bars are not likely to create very large internal movements: hence we expect each point to keep the same neighborhood throughout the process. Whether large displacements should happen, they are handled by retriangulating the current set of points so that subsequent algorithm iterations can still rely on a valid physical analogy of the distribution

as a truss structure. With such properties, the algorithm is likely to produce representations that retain some aspects of the analytic sound–control correspondence property.

- **step 3:** perform a Delaunay triangulation of the points distribution.

while !exit

- **step a:** update data point positions.
- **step b:** move points that went outside the user-defined region to the closest boundary point.
- **step c: if** all points have moved less than a significant distance, **exit = true**.
- **step d: if** points have moved more than the maximum allowed distance in respect to the previous triangulation, perform a Delaunay triangulation of the distribution.

Figure 4 shows various distributions obtained after applying the *unispring* algorithm to the representation shown in figure 1. They provide examples of square, circular and polygonal user-defined regions with uniform point densities, as well as non-uniform user-defined point density (shown in the case of a square region).

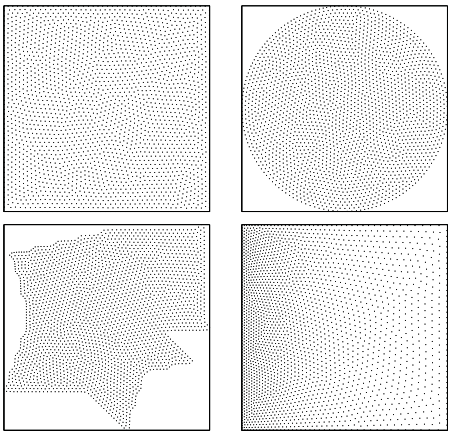

Figure 4: *Database representations obtained with the* unispring *algorithm. From left to right, top to bottom: square, circular and polygonal user-defined regions with uniform point densities; non-uniform user-defined point density within a square region.*

4. EVALUATION

4.1. Objectives

The aim of the evaluation process was to measure the algorithm performance relative to its expected features: redistribution of data points inside a user-defined region with internal user-defined density, and as much preservation as possible of the analytic sound–control correspondence property. Our methodology combines graphical results with quantitative measures.

Subjective visual appreciation of the algorithm action is a simple yet efficient way to make sure the final representation is constrained into the user-defined region. Evaluation was carried out using a square region.

The algorithm ability to produce a representation corresponding to the user-defined point density is evaluated in the standard case of a density-uniformisation task. The algorithm performance is assessed using a measure λ based on the normalized standard error of the 1-nearest-neighbors distance distribution [11]. For a set of N points $(P_i)_{1\leq i\leq N}$, we define λ by

$$\lambda = \frac{1}{\bar{\gamma}}\left(\frac{1}{N}\sum_{i=1}^{N}(\gamma_i - \bar{\gamma})^2\right)^{\frac{1}{2}}, \tag{2}$$

where

$$\gamma_i = \min_{j=1,\ldots,N, j\neq i} P_iP_j \text{ for } i = 1, ..., N$$

and

$$\bar{\gamma} = \frac{1}{N}\sum_{i=1}^{N}\gamma_i.$$

For a perfectly uniform point distribution, $\lambda = 0$; hence the smaller the value of λ, the more uniform the distribution. However, λ does not measure the algorithm performance in terms of constraining points into the user-defined region. Hence it is important to pair this measure with visual inspection of the final representation.

How the algorithm preserves analytic sound–control correspondence is estimated using three quantitative measures. To see if this property is altered, we look at pairs of points: an exception to the analytic sound–control correspondence property is introduced if at least one pair of points exchange their coordinate values on one or two axes after applying the algorithm. For instance, if points P_1 and P_2 have initial x-coordinate values such as $x_1 < x_2$, an exception is introduced if new coordinates are such that $x_1'' \geq x_2''$. The first two measures used for evaluation consist in the percentage p_1 of data point pairs that have exchanged coordinate values on one axis, and the percentage p_2 of pairs that have exchanged coordinate values on two axes. The third measure used for evaluation takes into account both p_1 and p_2 to compute a "distortion measure" value d that increases as the algorithm introduces more exceptions to the analytic sound–control correspondence property. More weight is given to pairs that have exchanged both coordinates. In our evaluation, we use

$$d = 5p_1 + 10p_2. \tag{3}$$

During the evaluation process, we refer to the pre-uniformisation steps of *unispring* (3.2) as the *uniform* algorithm, and the physical algorithm steps (3.3) as the *spring* algorithm. We compare these algorithms to the *unispring* algorithm, as they can be used independently for density uniformisation inside a square region (both were initially designed for that purpose [6]). By construction, the *uniform* algorithm fully preserves the analytic sound–control correspondence; but by doing so, we expect its performances in terms of density-uniformisation to be inferior to those of other algorithms. Evaluating the *spring* algorithm alone allows assessing the interest of the pre-uniformisation steps in *unispring*.

4.2. Evaluation Data

The evaluation was performed on five representations, providing different geometric configurations. They were obtained us-

ing CataRT sound-descriptors computation capabilities, and manual selection of two features.

- the *madeleine* representation (2404 points), with Spectral Centroid and Periodicity features, is made of subway sounds and presents a single-centered point distribution.
- the *gaussian* representation (2404 points), with 2-centered Gaussian artificial feature values on each axis, corresponds to a critical case with two very distinct centers.
- the *wave* representation (2404 points), with Start Time and Periodicity features, is made of environmental sounds and presents no identifiable distribution center.
- the *partita* representation (388 points), with Spectral Centroid and Note Number features, is made of instrumental sounds and presents initial coordinate values ranges dominated by the coordinate values of a few marginal points.
- the *alarm* representation (779 points), with Spectral Centroid and Note Number features, consists of different samples from the Freesound online library[3]. Points associated with the same MIDI notes appear as horizontal lines in the 2D display.

4.3. Graphical Results

Figures 5, 6, 7, 8 and 9 show the original evaluation representations and the final representations obtained by applying the *uniform*, *spring* and *unispring* algorithms. Unsurprisingly, the *uniform* algorithm alone is unable to fulfill the density-uniformisation task set by the user. The *spring* and *unispring* algorithms both seem to give satisfying results, providing square-shaped representations.

4.4. Quantitative Measures

Table 1 gives the values of the quantitative measures λ, p_1, p_2 and d defined in section 4.1.

By construction, the *uniform* algorithm fully preserves the analytic sound–control correspondence property ($d = 0$). However, this "zero-distortion" action prevents the algorithm from efficiently performing the tasks set by the user : besides failing at producing a square-shaped representation, this algorithm is responsible for higher λ values that any other algorithm, which underlines its lower performance in terms of density-uniformisation.

In contrast, *spring* and *unispring* are associated with lower and comparable λ values. With λ values respectively equal to 3.7 % and 3.8 % of the original representations' λ values on average, both algorithms successfully performed the density-uniformisation task. One necessary drawback to this is the introduction of distortion, though in different amounts. Since the absolute values of p_1, p_2 and d depend strongly on the initial representation geometry, algorithms have to be compared based on their action on a single representation. Reviewing the results representation by representation we see that the *unispring* algorithm provides lower values of d, which means that it is better at preserving analytic sound–control correspondence. This can also be noted by looking at percentages p_1 and p_2, which are lower in the case of *unispring*.

The mean values of p_1 and p_2 obtained with *unispring* are $\bar{p_1} = 24.13$ % and $\bar{p_2} = 0.49$ %. On average, a quarter of data point pairs have exhanged their coordinates on one axis, but the more preocuppying case of pairs having exchanged their coordinates on two axes only happened for a very few of them. Hence the algorithm performs well at preserving analytic sound–control correspondence. This property can be partly accounted for by the physical analogy on which *unispring* is based, as explained in section 3.3. As *spring* is also based on the same physical model, greater performances of *unispring* confirm the interest of its pre-uniformisation steps. They prevent the subsequent physical algorithm from having to process high-density regions of points, which are more likely to create anarchic movements during early iterations.

Pre-uniformisation is a much faster process than the physical algorithm, which gets iterated many times: consequently, the *spring* and *unispring* algorithms' speeds can be compared by this iteration count alone. Using this criteria, the *unispring* algorithm proved to be 1.2 times faster on average than *spring*. Taking only into account the number of retriangulations happening during the iteration process (cf. **step d** in section 3.3), *unispring* appeared to be twice faster on average than *spring*.

5. APPLICATIONS AND FUTURE WORK

5.1. Digital Musical Instrument Interfaces

Interactive corpus-based concatenative synthesis, combined with a two-dimensional single- or multi-touch control surface, can be used as a digital musical instrument (DMI) to "play" a sound database by navigation through the feature space. Here, our algorithm allows to exploit the totality of the interaction surface for control, while preserving the original analytic sound–control correspondence property and thus the perceptual meaning of the instrumental gestures.

The algorithm has been used to lay out more than 2500 sound segments for interaction on a multi-touch surface controlling the CataRT system[4] in the piece *Alarm–Signal* performed by Diemo Schwarz at the Sound and Music Computing Conference, Barcelona 2010.[5]

A video documentation can be found in the on-line journal *Musimédiane* [12].

5.2. Preset Interpolation

Another prospective application is the layout of many synthesiser or effect presets on a 2D plane for position-based preset interpolation, where the proximity of the cursor to the position of the closest presets determines their influence on the sound [13, 14] (see also *pMix*[6] for Max/MSP).

When a large number of presets is used, manual positioning might become unfeasable, and our automatic layout algorithm could again optimise the interaction space.

5.3. Comprehensive User Interface

The expected final point density has to be provided to the algorithm in the form of the desired length function $h(x, y)$. We studied several ways of making this process more user-friendly, with the aim of providing automatic calculation of h in the most common cases.

[3] `http://www.freesound.org`

[4] `http://imtr.ircam.fr/imtr/CataRT`

[5] `http://mtbf.concatenative.net`

[6] `http://www.olilarkin.co.uk/index.php?p=pmix`

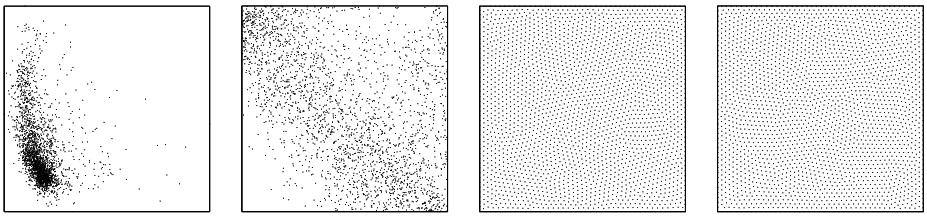

Figure 5: Madeleine *representation. From left to right: original representation,* uniform, spring, unispring *algorithm representations*.

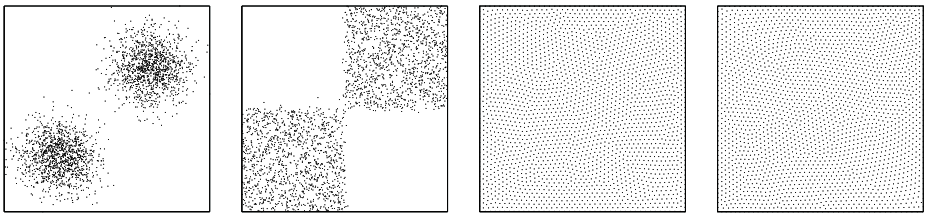

Figure 6: Gaussian *representation. From left to right: original representation,* uniform, spring, unispring *algorithm representations*.

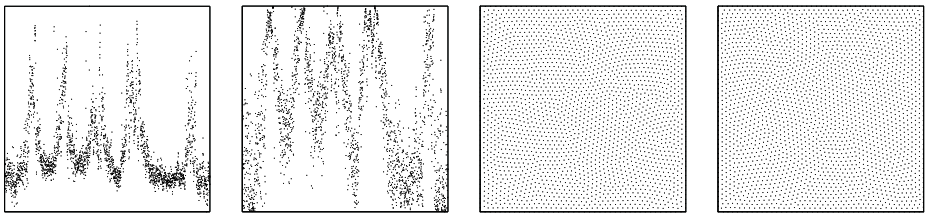

Figure 7: Wave *representation. From left to right: original representation,* uniform, spring, unispring *algorithm representations*.

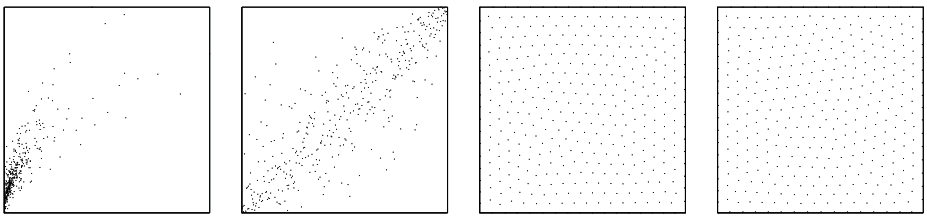

Figure 8: Partita *representation. From left to right: original representation,* uniform, spring, unispring *algorithm representations*.

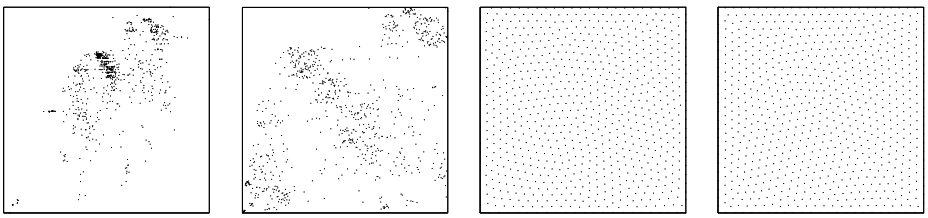

Figure 9: Alarm *representation. From left to right: original representation,* uniform, spring, unispring *algorithm representations*.

Algorithm	Measure	*madeleine*	*gaussian*	*wave*	*partita*	*alarm*
none (reference)	λ	1.7812	0.9876	0.9379	2.0258	1.3090
uniform	λ	0.6388	0.5288	0.7179	0.7994	0.8614
	p_1	0 %	0 %	0 %	0 %	0 %
	p_2	0 %	0 %	0 %	0 %	0 %
	d	0	0	0	0	0
spring	λ	0.0484	0.0503	0.0457	0.0518	0.0463
	p_1	41.8778 %	34.1827 %	27.4539 %	43.1718 %	28.2426 %
	p_2	9.8234 %	2.6994 %	2.4235 %	6.6142 %	2.8476 %
	d	3.0762	1.9791	1.6150	2.8200	1.6969
unispring	λ	0.0448	0.0487	0.0445	0.0482	0.0498
	p_1	21.6047 %	28.7839 %	16.6518 %	33.7426 %	19.8733 %
	p_2	0.54385 %	0.88407 %	0.46855 %	0.17227 %	0.36181 %
	d	1.1346	1.5276	0.8794	1.7044	1.0298

Table 1: *Values of quantitative measures λ, p_1, p_2 and d.*

For instance, partially-uniform representations can be obtained by using

$$h(x, y) = \frac{1}{d(x, y) + d_0},$$

where $d(x, y)$ is the positive original representation density (computed using kernel density estimation [15] for instance) and d_0 a constant used to modulate the amount of uniformisation. When d_0 becomes much greater than $d(x, y)$ for all x, y, h tends to a uniform length function providing total uniformisation (3.1). Figure 10 shows partially-uniform representations obtained with this method.

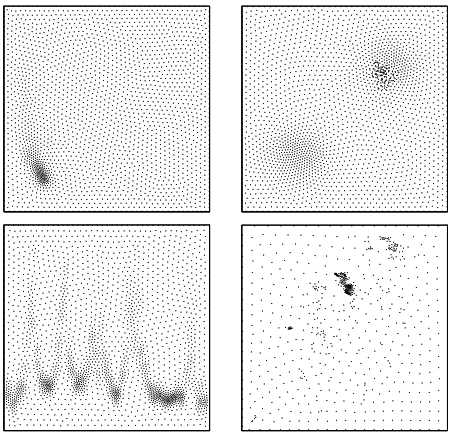

Figure 10: *Partially-uniform representations obtained by automatically computing* h *from the initial density. Original representations used (left to right, top to bottom):* madeleine *($d_0 = 5$),* gaussian *($d_0 = 2$),* wave *($d_0 = 1$),* alarm *($d_0 = 1$).*

5.4. Task-based User Evaluation

By providing a convenient, user-defined way to redistribute a database representation, our algorithm could possibly make sound design tasks easier. To test this hypothesis, we are currently considering conducting a task-based user evaluation, where users would be asked to execute basic sound-design actions by interacting with both transformed and original representations.

6. CONCLUSIONS

The *unispring* algorithm is an efficient solution to transform 2-dimensional database representations into user-specified representations. It is based on a physical algorithm whose performances are increased by pre-uniformisation of the data. The user defines a region inside which the algorithm redistributes the data points according to a user-defined length function, which determines the final distribution density.

Two-dimensional representations provide a convenient framework for interacting with sound, as in many contexts two degrees of freedom are used for interaction. Automatically extracting sound features from the database elements allows to create representations of large databases in which analytic correspondence between interaction control and evolution of sound characteristics can be found. With such representations it is often straightforward to create a mapping between the interaction setup and the sound data. However, it is unlikely that the user will consider this mapping optimal. Our algorithm allows the user to redistribute the data points in a way that will suit his needs more, while taking care of preserving as much as possible the analytic correspondence between interaction control and sound characteristics.

Our evaluation process provides information on how much this correspondence is altered by applying the algorithm. With a reasonable mean value of 24.13 % of data points pairs that have exchanged their coordinate values on one axis and only 0.49 % of pairs that have exchanged their coordinate values on two axes, the algorithm preserves most aspects of the analytic sound–control correspondence property. The algorithm shows excellent perfor-

mance when asked to perform the density-uniformisation task used for evaluation.

Unispring applications may include live performances, sound installations, sound design or educational courses. In order to make it easier for users to use the algorithm, we plan to release its real-time implementation in Max/MSP as an MnM object [16] by the time of the conference.

7. ACKNOWLEDGMENTS

The work presented here is partially funded by the *Agence Nationale de la Recherche* within the project *Topophonie*, ANR-09-CORD-022, `http://topophonie.fr`.

8. REFERENCES

[1] D. Schwarz, "Corpus-based concatenative synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 92–104, 2007, Special Section: Signal Processing for Sound Synthesis.

[2] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the Cuidado project," Tech. Rep. version 1.0, Ircam – Centre Pompidou, Paris, France, Apr. 2004.

[3] D. Schwarz, R. Cahen, and S. Britton, "Principles and applications of interactive corpus-based concatenative synthesis," in *Journées d'Informatique Musicale (JIM)*, GMEA, Albi, France, Mar. 2008.

[4] S. Jordà, "On stage: the reactable and other musical tangibles go real," *International Journal of Arts and Technology*, vol. 1, pp. 268–287, 2008.

[5] D. Schwarz and N. Schnell, "Sound search by content-based navigation in large databases," in *Proceedings of 6th Sound and Music Computing Conference (SMC'09)*, Porto, Portugal, Jul. 2009.

[6] I. Lallemand, "Optimized visualisation for navigation in large sound databases," M.S. thesis, Université Pierre et Marie Curie, Sep. 2010.

[7] J. Shlens, "A tutorial on principal component analysis," Dec. 2005.

[8] D. Skočaj, A. Leonardis, and H. Bischof, "Weighted and robust learning of subspace representations," *Pattern Recogn.*, vol. 40, no. 5, pp. 1556–1569, 2007.

[9] P. Persson and G. Strang, "A simple mesh generator in Matlab," *SIAM Review*, vol. 46, pp. 2004, 2004.

[10] I. M. Mitchell, "The flexible, extensible and efficient toolbox of level set methods," *J. Sci. Comput.*, vol. 35, no. 2-3, pp. 300–329, 2008.

[11] H. Nguyen and J. Burkardt and M. Gunzburger and L. Ju and Y. Saka, "Constrained CVT meshes and a comparison of triangular mesh generators," *Computational Geometry*, vol. 42, no. 1, pp. 1 – 19, 2009.

[12] A. Bonardi, F. Rousseaux, D. Schwarz, and B. Roadley, "La collection numérique comme paradigme de synthèse/composition interactive," *Musimédiane: Revue Audiovisuelle et Multimédia d'Analyse Musicale*, , no. 6, 2011, `http://www.musimediane.com/numero6/COLLECTIONS/`.

[13] A. Momeni and D. Wessel, "Characterizing and controlling musical material intuitively with geometric models," in *Proceedings of the 3rd International Conference on New Interfaces for Musical Expression (NIME'03)*, Montréal, Canada, May 2003, pp. 54–62, National University of Singapore.

[14] A. Freed, J. MacCallum, A. Schmeder, and D. Wessel, "Visualizations and Interaction Strategies for Hybridization Interfaces," in *Proceedings of the 10th International Conference on New Interfaces for Musical Expression (NIME'10)*, 2010, pp. 343–347.

[15] B. W. Silverman, *Density estimation: for statistics and data analysis*, Chapman and Hall, London, 1986.

[16] F. Bevilacqua, R. Müller, and N. Schnell, "MnM: a Max/MSP mapping toolbox," in *Proceedings of the 5th International Conference on New Interfaces for Musical Expression (NIME'05)*, Singapore, Singapore, May 2005, pp. 85–88, National University of Singapore.