Optimised visualisation for navigation in large sound databases

Ianis Lallemand

Internship report - ATIAM master's degree

March 1st, 2010 - June 30th, 2010

Ircam, IMTR Team, under the supervision of Diemo Schwarz

Abstract

During this internship, which took place in the Real-Time Musical Interactions Team (Ircam) under the supervisation of Diemo Schwarz, I studied the problem of developing visualisation optimisation algorithms. This work was motivated by the resolution and performance issues that could arise using CataRT, a corpus-based concatenative synthesis software that allows the user to select short snippets of sound through their position in a 2D projection of a high-dimensional signal descriptor space, in musical or professional applications. We first chose to perform the optimisation tasks inside a pre-selected 2D projection, thus taking a different approach than dimensionality reduction. With the idea of preserving the physical meaning of the coordinate system axes in mind, I then developed a range of algorithms that perform geometrical transformations on a two-dimensional set of points in order to spread it as uniformly as possible across a chosen enclosing shape (a square, a disk or a polygon). I found that efficient uniformisation was not possible without introducing distortion (in a sense that will be defined in the report) in the point set, for which I developed an estimation algorithm. Evaluation of the five most efficient algorithms were performed on five different geometries, at the end of which two algorithms clearly stood out: the *domains* algorithm, a fast, low-distortion introducing algorithm that provide reasonable uniformisation, and the *uniform2D/delaunay* algorithm, a hybrid algorithm using a physical model to produce perfectly equally-spaced points distribution with reasonable distortion.

Contents

Acknowledgements

I would like to thank my internship supervisor, Diemo Schwarz, for offering me the opportunity to work on this subject with great freedom. I am grateful to him for being attentive to my ideas, as well as taking time to help me improve some of them. His interest in the outcomes of the internship has been a real source of motivation.

I would also like to thank the members of the IMTR team, for showing interest in my work and sharing their time and knowledge. I thank them for the friendly atmosphere they allowed me to work in.

Lastly, I would like to thank the interns I had the chance to meet — or meet again — in the course of these months at Ircam, for their friendliness and sense of humor.

Introduction

Ever-larger sound databases, nowadays easily available on the Internet or in sound banks, represent a great potential for musical creativity. Yet most of this potential may remain hidden to the user without efficient means of exploring the data: in 2005, for instance, the total duration of the Creative Commons sound archive *archive.org* could be estimated to be somewhere between 34.2 and 2,000 years [2], and has been increasing ever since. Consequently, methods focusing on content-based music information retrieval have attracted much attention in the past years, as they allow greater insight on the contents of the soundfiles (thanks to the use of audio descriptors) than usual keywords classification.

We focus here on corpus-based concatenative synthesis methods that make use of the descriptors approach to address the problems of creative use and interactive exploration of large-scale databases. They combine sound segmentation and descriptor analysis of selected sound files (the *corpus*) to allow their exploration in a non-linear temporal order. Small sound segments (refered to as *units*) can be displayed according to their positions in the descriptor space, thus allowing the user to interactively select segments according to their sonic contents. Applications include musical composition, sound design and interactive exploration of the corpus, but also live performances (thinking of the corpus and its representation as an instrument) and audio installations.

The results obtained during this internship can be applied to CataRT, a corpus-based concatenative synthesis software developped by Diemo Schwarz at Ircam (Paris), which allows to plot the units according to the values of two chosen audio descriptors. The corpus can the be explored using an XY controller or a multitouch surface, or mapped on a physical surface (for sound installations), allowing the audience to play sounds according to their positions. However, it is common that the units form an aggregate in a small region of the plane, with a few isolated units occupying distant positions. This configuration leave "blanks" in the representation space, i.e. regions where the units' density is very low compared to that in the aggregate. In such cases, this "scientific" plot, though providing crucial informations about the units mutual hierarchy in terms of descriptor values, doesn't optimise the interaction space available to the user: most units are located in a small area, and most of the space is not associated with any unit. This is particularly annoying for music interaction and sound installations, where one might want to avoid "silent" zones at all costs. The *Topophonie* ANR-project[1], whose applications include integrating sounds in 3-dimensional computer maps — for instance by mapping a corpus onto them —, is another situation where silent zones have to be avoided.

The aim of this internship was to develop Matlab algorithms that, through geometrical transformation of the corpus representation, would optimise the use of the interaction space. A key aspect in this task was to keep in mind the hierarchical relations between the units implied by the descriptor values, and try to preserve them as much as possible throughout the transformation; consequently, the distortion introduced by the algorithms had to be defined and measured. This report first gives a few elements about corpus-based concatenative syn-

[1] "The research project Topophonie proposes lines of research and innovative developments for sound and visual navigation in spaces composed of multiple and disseminated sound and visual elements." (http://www.topophonie.fr)

thesis and defines more precisely the context in which the aforementioned problem is defined. Following a bibliographical study, the algorithms are presented according to the dimension of the underlying geometric transformations: one-dimensional, pseudo two-dimensional and two-dimensional. Finally, the distortion measure is defined and the five most relevant algorithms are evaluated on selected corpora.

1 Interests and context of the internship

1.1 An overview of corpus-based concatenative synthesis

Corpus-based concatenative synthesis is a form of concatenative synthesis, which has been in use in speech synthesis for the past fifteen years. It differs from usual synthesis methods in the sense that it does not require to build a model of the sound signal that would realistically generate all of its fine details, which is a very difficult task. Instead, actual recordings are used to achieve the synthesis expressivity.

Corpus-based approaches are more specific than *concatenative synthesis* methods, which refers to using segments of sound data to assemble the resulting sound. The term *corpus* refers to methods which selects sound segments automatically out of a chosen database (the *corpus*), which differs from using a small number of segments with a predetermined selection [13]. Hence, in corpus-based synthesis, the synthesis expressivity and the corpus size are directly related.

CataRT is an application of corpus-based concatenative methods[2]. It combines audio descriptor analysis of the units, extracted from the corpus with a typical length of a few hundreds millisecond[3], with direct selection in real-time thanks to a plot of two chosen descriptor values. The user interface (cf. Fig. 1) offers different ways of selecting the units, as well as audio transformations. Descriptor analysis is a key aspect of the approach: as the corpus size increases (which increases the synthesis expressivity), it is necessary to classify the units according to quantitative measurements that give a proper insight on their sonic properties. Two major kinds of descriptors are used in CataRT: segment descriptors, which describe the units themselves (such as Start Time, Duration, ...), and content descriptors, which describe the audio content of the units. They are the unit's average of the instantaneous descriptor values that are calculated each 40 ms. In this report, I chose to use the following descriptors in the corpus examples: *Spectral Centroid*, *Periodicity*, *Note Number*, *Start Time* and *Unit ID*.

- 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;
- the *Periodicity* measures the harmonic character against the noisy character of the sound;
- the *Note Number* is the fundamental frequency of the unit in fractional MIDI notation;
- the *Start Time* is the start time of the unit in the sound file in ms;
- the *Unit ID* is the unit unique index.

[2] for an extensive and historical review of the existing approaches, see [12].

[3] CataRT features different segmentation methods: in this report, I used the *chop* mode, which segments audio into equal-sized units every given ms, and the *Yin note segmentation*, which segments audio per change of pitch.

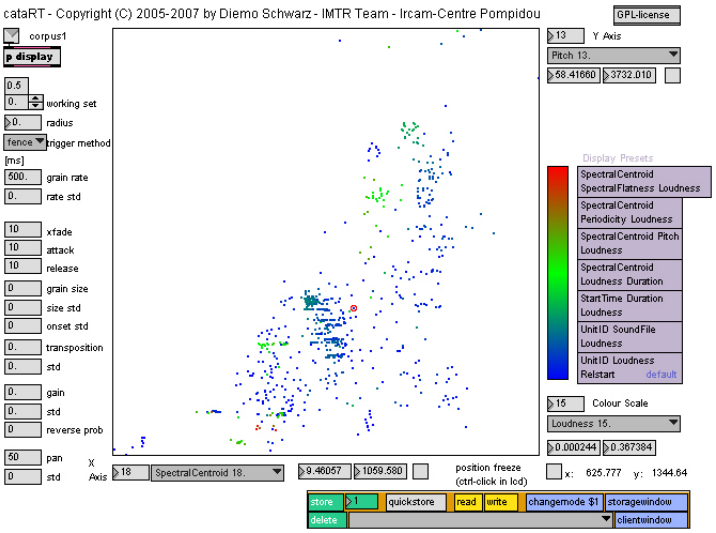

Figure 1: CataRT user interface.

1.2 Applications and restrictions due to current visualisation methods

Audio descriptor analysis and real-time unit selection allows different kinds of applications. A first example is interactive exploration of the corpus, whose aim might be to look for a specific event in a large database. One can imagine, for instance, a sound designer looking for a specific sound in a sound bank, or trying to locate glitches in audio recordings (which can be easily located with the use of the proper descriptors, as their audio characteristics are radically different from those of the "normal" bits of the recordings).

Another example is the creative use of a selected corpus and of the synthesis capabilities of the software. The corpus can be "played" by navigating through the descriptor space, selecting units with a physical interface such as the computer mouse, a XY controller or a multitouch surface. This can be described by the term *free synthesis*, and can take place in the composition or sound design process, i.e. in the studio, or in the context of live performances.

Finally, the link between sound and a 2-dimensional surface that the descriptor plot creates can be used in sound installations, where a corpus can be mapped onto a surface, thus allowing the audience to play sound as it touches the surface or moves across it. This kind of mapping can also be made virtually: the ANR-project *Topophonie* aims to integrate sounds in 3-dimensional computer maps, to recreate ambiances and augment the user experience.

All of these applications rely on a visualisation of the corpus, that CataRT provides through the plot of two descriptors. In a field recording case for instance (a typical *corpus* example), most units will be sonically close to each other, with only a few distinct events happening from time to time in the recording. When plotting their positions in a "scientific" way, i.e. giving each unit the precise position it has according to the value of the chosen descriptors, it is very likely that most units will be part of an aggregate. The remaining units, having marginal descriptor values, will be located outside these high-density areas. The density non-uniformity may pose different problems to the user interested in aforementioned applications.

First of all, the fact that most units are located in a small area compared to that occupied by the whole unit set poses a resolution problem. One can not simultaneously interact with

units in an aggregate and marginal units with the same precision amount; in fact, it is common that one cannot visually differentiate the units in the aggregate one from another, due to the great number of segments involved.

As shown in figure 2, where a XY controller with superimposed units position can be seen, another problem is that low density regions cause blank areas to occupy most of the interaction space. This might be a problem in the case of a multiple-corpus based live performance without visual feedback (which is not available on most XY controllers, to the exception of multitouch screens), in which it would not be possible to memorize the units' positions for each corpus, as well as in the case of sound installations (and *Topophonie* applications) where the blank areas would be unwanted "silent" zones.

Figure 2: XY controller with superimposed units.

1.3 Chosen approach and objectives

According to Schneiderman's information seeking mantra [19], a solution to the resolution problem might be to filter the amount of units to display, then to allow the user to zoom inside the aggregate for improved resolution. Though this solution might present advantages of its own, I focused on another approach during this internship, due to the contexts in which interactions with the corpus might take place.

In the case of *free synthesis* for instance, it seems natural to interact with the corpus using a XY square-shaped controller, as the units are plotted by considering the chosen descriptors as orthonormal coordinates. Although multitouch display surfaces have become more widely available recently, most XY controllers don't offer a visual feedback of what is happening on the computer screen. In this context, as well as in the case of a sound installation in which a corpus is mapped onto a physical space, it seems more appropriate to adopt a static mapping between the corpus and the XY surface. This defines one of the first objectives of the algorithms that I had to develop in this internship: to maintain simultaneous access to all of the units in a selected enclosed space, while improving separability of the units which are initially very close to each other. As most controllers are square-shaped, I chose the square as the main interaction space example; however, as sound installations and *Topophonie* applications may benefit for other kinds of interaction space, I provided solutions for disk-shaped and polygon-shaped spaces.

Being given an enclosing interaction space, this first goal naturally translates in terms of units density uniformisation. In a sense, spreading the units through the space to reduce the

density non-uniformities corresponds to an optimal use of the interaction space : in the ideal case of equally-spaced units, each unit will dispose of the same area for its selection.

The high-dimensional context of the descriptor space, and the fact that only two descriptors are used for visualisation, suggest that dimensionality reduction methods such as PCA might be helpful in finding a convenient corpus visualisation. However, the new coordinate system provided by PCA does not have physical meaning, as opposed to the original descriptor coordinate system. In this internship, we consequently chose to perform a geometric transformation of an existing projection defined by two selected descriptors, so that the axis used in the optimised visualisation are still associated with clearly identified audio properties. This approach, which focuses on preserving one of the strengths of the CataRT synthesis method (the audio descriptor analysis and quantitative classification of audio units), would not be consistent if the original hierarchical relations between units, given by the descriptor values, were not preserved as much as possible throughout the transformation. This constraint constituted the objectives of the algorithms: to introduce as little distortion in the corpus as possible, in a sense that had to be defined. Hence a distortion measure algorithm had also to be developed (detailed in 6.1).

2 Bibliographical study

2.1 Data visualisation methods

As units are represented as a point cloud in CataRT, data visualisation methods are natural possible bibliographical references. However, I found that most methods followed the previously mentioned Schneiderman's information seeking mantra [19], for instance in the context of providing efficient visualisation methods in small screen displays [1] (whose sizes are not far from those of most XY controllers). According to Schneiderman, the visualisation must support the following kinds of interaction: *details on demand*, *zoom* and *filter*; two of them are related to our visualisation context:

- *zoom*: data overlapping due to high concentration of units in small areas prevent the user to accurately select a particular unit, a task made possible by zooming close enough in the high-density region. However, for the reasons explained in 1.3, this approach was not studied during this internship;
- *filter*: this method aims at preventing data overlapping by removing certain units from the visualisation. On the contrary, the approach I followed during this internship tries to preserve simultaneous access to all of the units.

Although not corresponding to my objectives during this internship, it has to be noted that these methods might be useful for adding multiple-scale representations to CataRT display - a task that might be of interest in the case of very large corpora. Examples of advanced zoom–based approached can be found in [21] and [22], where multiple "fish-eye" lenses are combined with dimensionality reduction methods.

The algorithms developed during this internship differ from most data visualisation methods in the sense that, by performing geometrical transformations over units positions, they

alter the descriptor value and hence, the data itself. Consequently, our approach seems closer to the field of data normalisation than to that of data visualisation.

2.2 Character normalisation methods

Relevant algorithms were found in the field of character normalisation for recognition of handwritten letters. I found that the first algorithms described in [24] could be related to the *uniform2D* algorithm I had developed soon after the beginning of the internship. However, character normalisation for letters recognition aims at preserving the overall shape of the letter, and the article thus describes improvements that no longer correspond to the internship context.

2.3 Mass spring damper algorithms

As detailed in 5.2, one of the most efficient algorithm developed during this internship is based on a physical mass spring damper system. Mass spring damper-based algorithms had previously been studied by Diemo Schwarz and Norbert Schnell at the IMTR team [17]. A mass spring damper system was first applied to an existing projection (which is the chosen context in this internship) to avoid overlapping point by pushing them appart (cf. Fig. 3). This algorithm shares a similar physical model to that of the *delaunay* algorithm detailed in 5.2, although the algorithm developed during this internship was not inspired by this one but by that of the *distmesh* toolbox (as detailed in 5.1 and 5.2). The main mass spring damper-relating result in [17] is a dimensionality reduction algorithm, which puts it out of the chosen context for the internship. However, the comparison between dimensionality reduction approaches and geometrical transformations of an existing projection, as well as the combined use of these two methods, are essential issues that will have to be studied in the future.

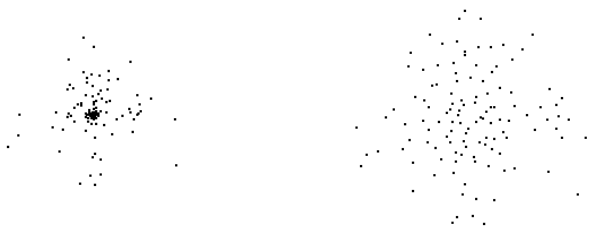

Figure 3: Effect of repulsion algorithm detailed in [17] (right) on a cluster (left).

3 One-dimensional algorithms

3.1 First constraint model: conservation of units order on each axis

As a key constraint that algorithms had to fulfill as much as possible, “preserving the hierarchical relations between units in the chosen projection” had first to be defined more precisely. It appeared quite naturally that an ideal transformation in regards to this constraint would preserve the units’ order on each axis. I defined this “zero-distortion” case by considering that the only unwanted effect of a geometrical transformation would be to modify the relative positions of units on any of the two descriptor axes: as the approach of working on an existing

2-dimensional projection of the high-dimensional descriptor space was motivated by the will to preserve the physical meaning of the coordinate system, it is necessary that an increase in unit positions along each axis still corresponds after the transformation to an increase in the corresponding descriptor values.

The aspect of the "hierarchical relations between units" I didn't chose to take into account in the definition of the zero-distortion case is how their descriptor values compare to each other: for instance, an algorithm that brings close to each other two originally distant units can still introduce no distortion as long as the units order on each axis is preserved. When using the transformed units representation, the user might consequently experience the expected increase in descriptor values as he moves towards higher positions in the coordinate system, but might not be able to predict how much increase in these values a displacement may bring. This choice was motivated by the very aim of the algorithms: to normalise the units density in a selected enclosing shape, which would result in equally-spaced units. However, preserving the original distance between units[4] might be desirable in some situations, and the objectives in the context of this internship should be considered as the first step towards more configurable transformations.

3.2 A simple 1D algorithm: the *uniform* algorithm

This one-dimensional algorithm was the first to be developed: it is a one-dimensional realisation of the two objectives that algorithms were suppose to fulfill. The reason I chose to consider a 1-dimensional algorithm, i.e. an algorithm that would work on the values of only one descriptor, is that an intuitive notion of order can only be found in 1D. Using 2-dimensional orders such as lexicographical order would assume that one of the two descriptors has more importance than the other one, which obviously does not suit the idea behind CataRT visualisation. For this reason it seemed natural to study the behavior of 1-dimensional algorithms, which could then be applied independently on each descriptor values. Although one might expect this approach to provide limited results due to its simplistic nature, it has the advantage of easily providing examples of zero-distortion algorithm.

The *uniform* algorithm normalises the units density of a single descriptor while preserving the units order. It works in three steps:

- **step 1:** units are sorted and their position in the sorted list is given as a function of their position in the unsorted list. For the unit occupying the position i in the unsorted list, the position in the sorted list is $n(i)$;
- **step 2:** the output list is filled in the same order than the original unsorted list. For each position i, a descriptor value of $n(i)$ is given;
- **step 3:** the ouptut list values are normalised between 0 and 1.

Choosing units order indexes as the new descriptor value is the simplest way to ensure density uniformisation and conservation of units order throughout the transformation. The action of the *uniform* algorithm was first observed on randomly generated Gaussian distributed values (cf. Fig. 4). Theses values are plotted on the x-axis, while Unit ID descriptor values (which are uniformly distributed by definition) are used on the y-axis for convenience.

[4]Either in the selected projection or in the high-dimensional descriptor space (cf. 6.4).

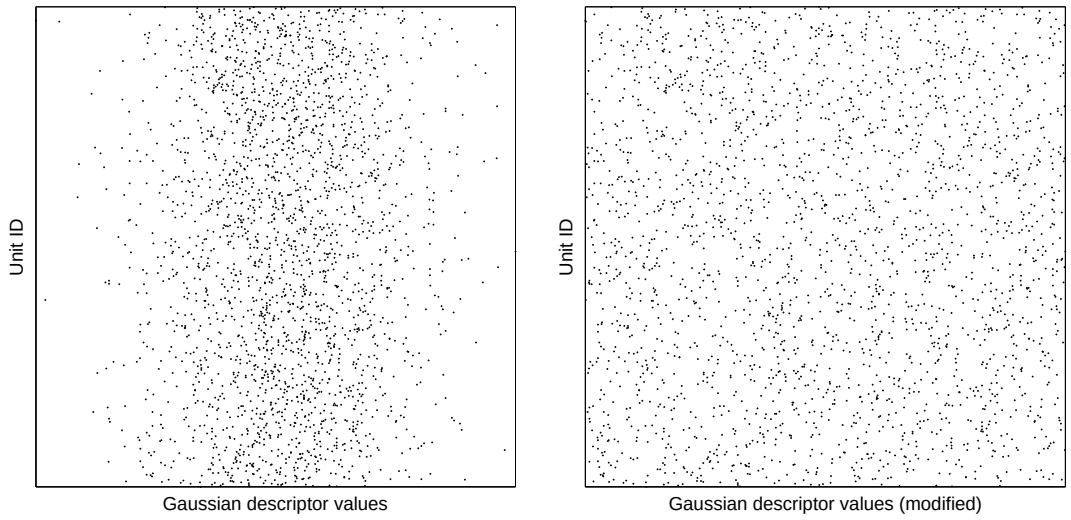

Figure 4: Action of the *uniform* algorithm on Gaussian distributed units. Left: before transformation; right: after transformation.

Figure 4 shows encouraging results: although the algorithm acts on the x-axis without taking into account the y-axis descriptor values (which explains why it does not produce equally-spaced units), it manages quite well to spread the units inside the enclosing square. However, as figure 5 shows it, this appears to be largely due to the simple nature of the Gaussian distribution used in figure 4. Using descriptor values obtained from audio analysis of an actual recording (the *madeleine* corpus), the limits of not taking into account the y-axis descriptor values are clearly apparent: although descriptor values are equally spaced on each axis (by definition of the Unit ID descriptor and thanks to the action of the *uniform* algorithm on the Spectral Centroid descriptor), units are far from being uniformly distributed in their 2-dimensional representation.

3.3 Mathematical formulation of the *uniform* algorithm

The *uniform* algorithm can also be defined by finding a mathematical function which would give the new descriptor values in function of the original ones. In order to preserve the values order throughout the transformation, this has to be a strictly increasing function. It seemed natural to try to compute this function from an evaluation of the units 1D-density (using only one descriptor values).

The density estimation is carried on by the *KDE* Matlab toolbox[5], which implements Kernel Density Estimation [20]. Kernel Density Estimation provides an x-axis estimate $\hat{f}$ of the density function f in the form of a sum of variations of a kernel function K:

$$\hat{f} = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)$$

where n is the number of kernels used for estimation, X_i the center of Kernel n°i and h a smoothing parameter called the *bandwidth*. The *KDE* toolbox determines the optimal number

[5]http://www.mathworks.com/matlabcentral/fileexchange/14034-kernel-density-estimator

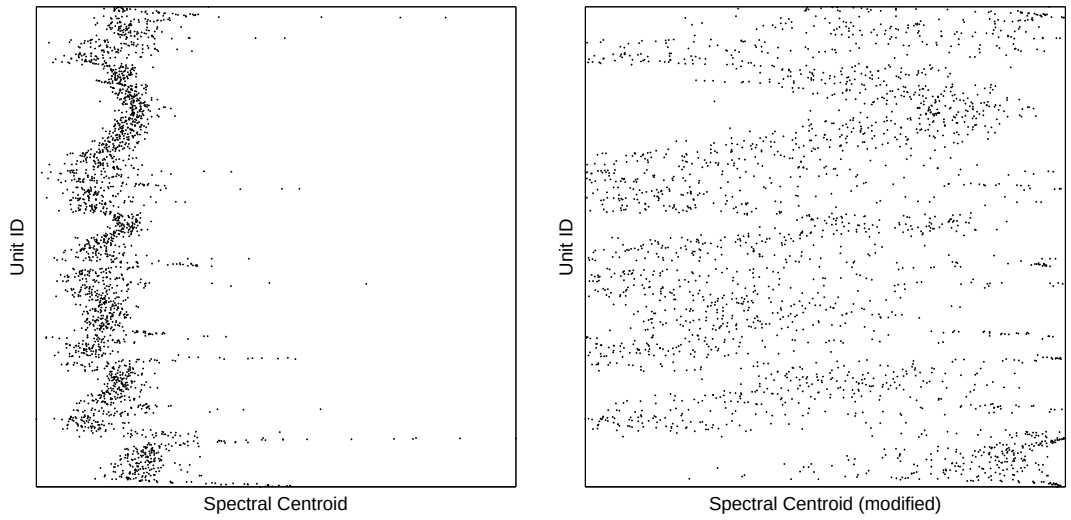

Figure 5: Action of the *uniform* algorithm on Spectral Centroid descriptor values (*madeleine* corpus). Left: before transformation; right: after transformation.

and centers positions of kernels (parameters n and X_i) and the optimal bandwidth, using a Gaussian kernel function with mean zero and variance 1. This kernel function shows clearly the effect of the h parameter:

$$K\left(\frac{x-X_i}{h}\right) = \frac{1}{\sqrt{2\pi}}\exp\left(\frac{(x-X_i)^2}{2h^2}\right)$$

hence the variance is controlled indirectly through h.

Firsts tests suggested that the distribution function F (the primitive of the density function, cf. Fig 6) provided the same results as the *uniform* algorithm, i.e. that the output values were uniformly distributed; I later found that this intuition was mathematically exact. Denoting as X the random variable that gives the x-axis descriptor values, the modified descriptor values are given by the random variable $F(X)$. Considering the probability for $F(X)$ to take a value under y, we have:

$$\begin{aligned} P\left(F(X) < y\right) &= \int_{\mathbb{R}} \mathbf{1}\left(F(x) < y\right) f(x)\mathrm{d}x \\ &= \int_{-\infty}^{F^{-1}(y)} f(x)\mathrm{d}x \\ &= F\left(F^{-1}(y)\right) \\ &= y \end{aligned}$$

which shows that F(X) is uniformly distributed. Although not crucial to the *uniform* algorithm, this mathematical formulation allows to express it in a more general way and will be used in improved algorithms.

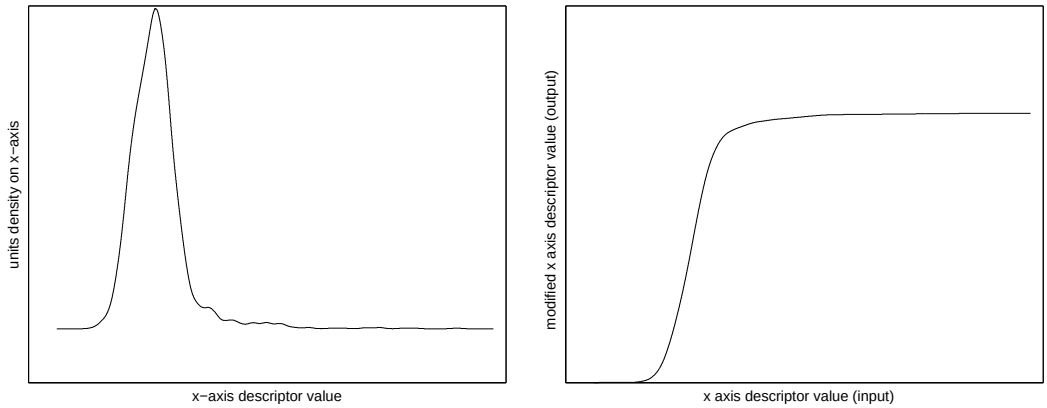

Figure 6: Units density f on x-axis (left) and corresponding primitive F (right).

3.4 A 2D extension: the *uniform2D* algorithm

Previous examples consisted in applying the *uniform* algorithm on a single descriptor, while setting the second descriptor to Unit ID for visualisation convenience. The *uniform2D* algorithm is the simplest possible extension of the *uniform* algorithm to the case of two descriptors: it simply consists in applying independent *uniform* algorithms on each axis. Predictably, the independent actions of the algorithms, while ensuring uniform repartition of the units projections on each axis, does not result in equally-spaced units in the 2D plane. Figure 7 shows the action of the *uniform2D* algorithm on Spectral Centroid and Periodicity descriptor values of the *madeleine* corpus. The independent action of the algorithm on each axis is particularly visible there, since the bottom left part of the square remains blank: this is because the units that had the lowest x-axis descriptor values were located in the upper half of the y-axis descriptor values prior to the transformation.

It is then easy to imagine a worst-case scenario for this algorithm: as figure 8 shows it, Gaussian descriptor values with two gaussian centers located at the same position on each axis result in a critical situation where the algorithm cannot spread the units of the bottom left gaussian center to the entire square domain, as (to the exception of a few units) their coordinates have lower values than those of any unit in the top right center.

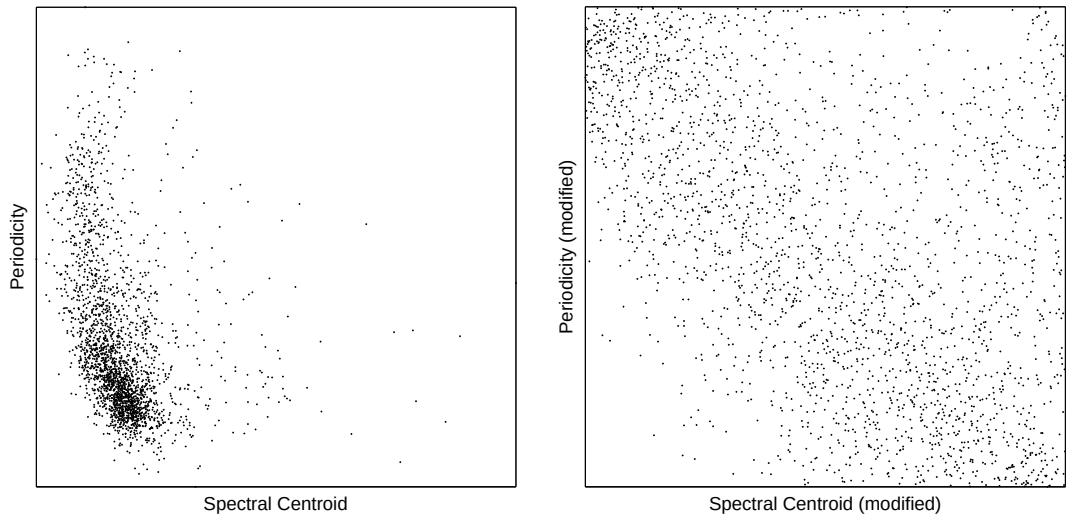

Figure 7: Action of the *uniform2D* algorithm on Spectral Centroid and Periodicity descriptor values (*madeleine* corpus). Left: before transformation; right: after transformation.

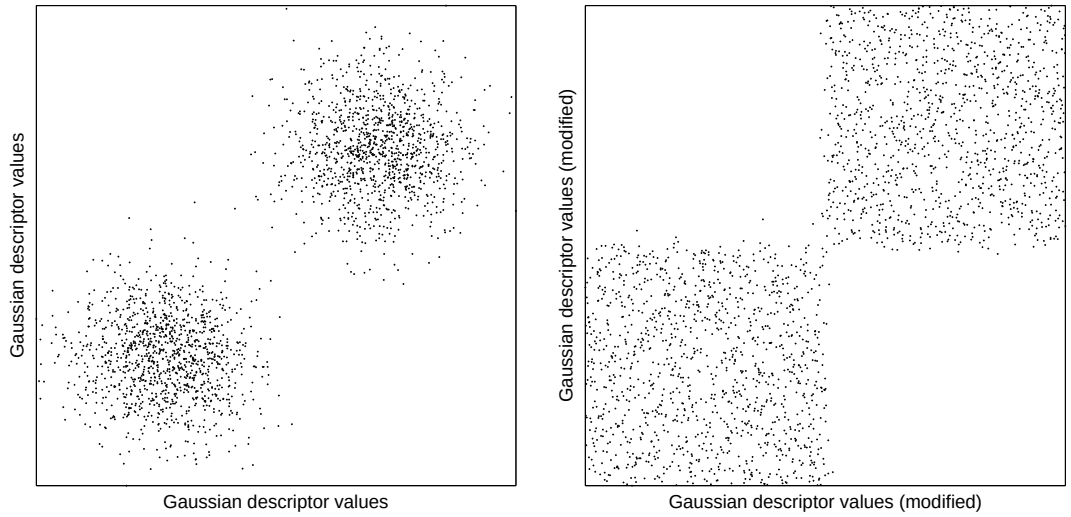

Figure 8: Action of the *uniform2D* algorithm on Gaussian descriptor values. Left: before transformation; right: after transformation.

4 Pseudo two-dimensional and two-dimensional algorithms

4.1 The *domains* algorithm

The situation depicted in figure 8 might inspire a way of improving the *uniform2D* algorithm. As illustrated by figure 9, one might want to move the transformed units to the blank zones of the enclosing square. However, this poses two distinct problems.

First of all, this further transformation of the descriptor values would break the conservation

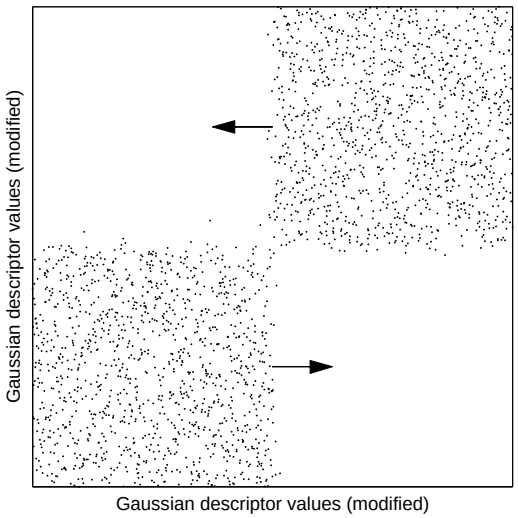

Figure 9: Modified Gaussian descriptor values and expected displacements for algorithm improvement.

of units order on each axis, hence introducing distortion. Yet it is clear that the zero-distortion constraint can only result in situations similar to that of figure 8, and has to be relaxed to improve the units spread across the square. As distortion should still remain as low as possible, this statement stresses the need of a distortion measure algorithm, which will be detailed in 26.

The second problem is that the further displacements of the units symbolised by the arrows in figure 9 favorises the x-axis over the y-axis, which is not conceptually satisfying. Due to the symmetry of this example, it is clear that horizontals displacements were here chosen arbitrarily and had no reasons to be favored over vertical arrows. However, even in cases were a direction might appear more suited to the geometry of the units set than the other one, an algorithm based on the currently discussed improvement of *uniform2D* will only be a *pseudo*-2-dimensional algorithm.

Although it was clear that true 2-dimensional algorithms would be more satisfying, I developed the *domains* algorithm based on the idea of a pseudo-2D algorithm that would allow some distortion to be introduced through the transformation. This algorithm was built with the worst-case scenario of the Gaussian descriptor values (from now on, this term will refer to the situation shown in the left part of figure 8, with two Gaussian centers on each axis) in mind. It works according to the following three steps:

- **step 1:** the *uniform* algorithm is applied on y-axis descriptor values, and resulting values normalised between 0 and 1;
- **step 2:** equally-wide domains are defined on the y-axis. Inside each domain i, the x-axis density of the enclosed points (i.e. the density of their projection on the x-axis) is estimated using Kernel Density Estimation, and the corresponding distribution function F_i is computed and normalised between 0 and 1;
- **step 3:** independent *uniform* algorithms are applied in each domain using the distribution functions. To limit distortion due to abrupt changes in x-axis displacements between two units having close y-axis descriptor values but belonging to different domains, the

x-axis output position for a given unit is interpolated. A unit of x-axis coordinate x in domain i gets $0.5F_{i-1}(x) + 0.5F_i(x)$ as a new x-axis coordinate if it is located on the lower boundary of i, and $0.5F_i(x) + 0.5F_{i+1}(x)$ if its is located on the upper boundary. X-axis output coordinates are linearly interpolated between these two values according to the y-axis position of the unit inside i, so that a unit in the middle of i gets $F_i(x)$ as a new x-axis coordinate[6].

These steps are summed up in figure 10, as well as expected displacements inside each domain. No arrow is shown in the middle domain since it is hard to predict the effect of the algorithm in this area which encloses units from both Gaussian centers; for this reason, it seemed preferable to use more than five domains in practice. The action of the algorithm was observed on the Gaussian descriptor values that inspired its design, using ten domains (a number which then proved to work well with all the corpora the algorithm was tried on). As figure 11 shows it, the algorithm performs quite well with Gaussian descriptor values.

Figure 12 shows the action of the *domains* algorithm on Spectral Centroid and Periodicity descriptor values from the *madeleine* corpus. In this context, the algorithm performs quite well, as units are efficiently spread across the whole square. However, this statement must be tempered by the measure of the distortion introduced by the algorithm, which will be discussed in 6.3.

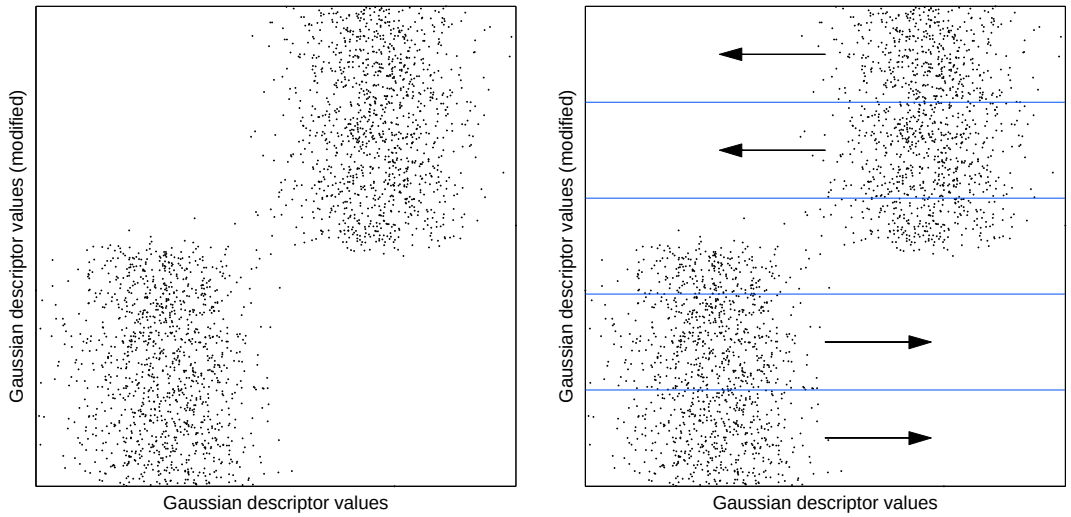

Figure 10: The *domains* algorithm. First step (left): the *uniform* algorithm is applied on y-axis descriptor values; second and third steps (right): independent *uniform* algorithms are applied on x-axis descriptor values inside each domain (delimited by blue lines).

[6]No interpolation takes place in the lower half of the first domain and the upper half of the last domain.

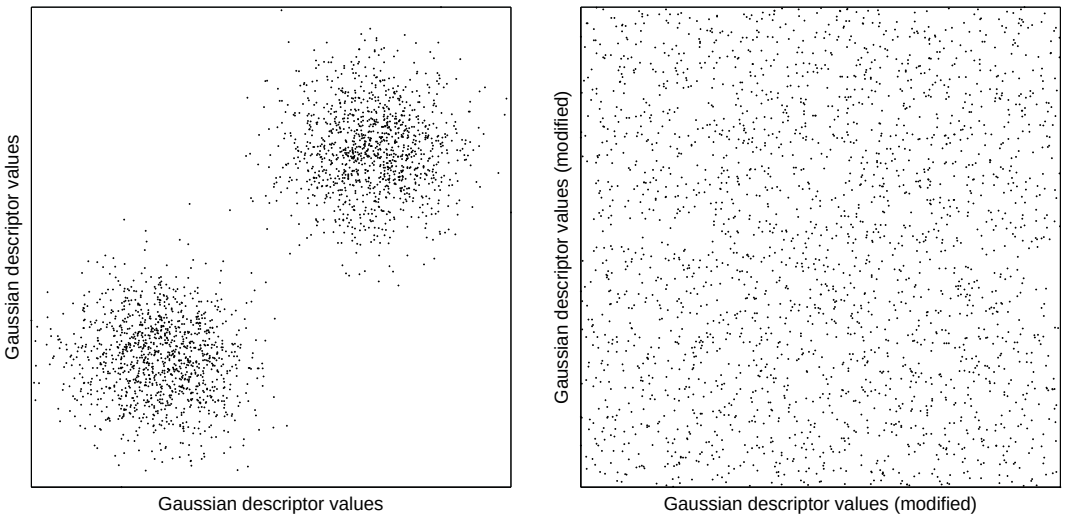

Figure 11: Action of the *domains* algorithm on Gaussian descriptor values. Left: before transformation; right: after transformation.

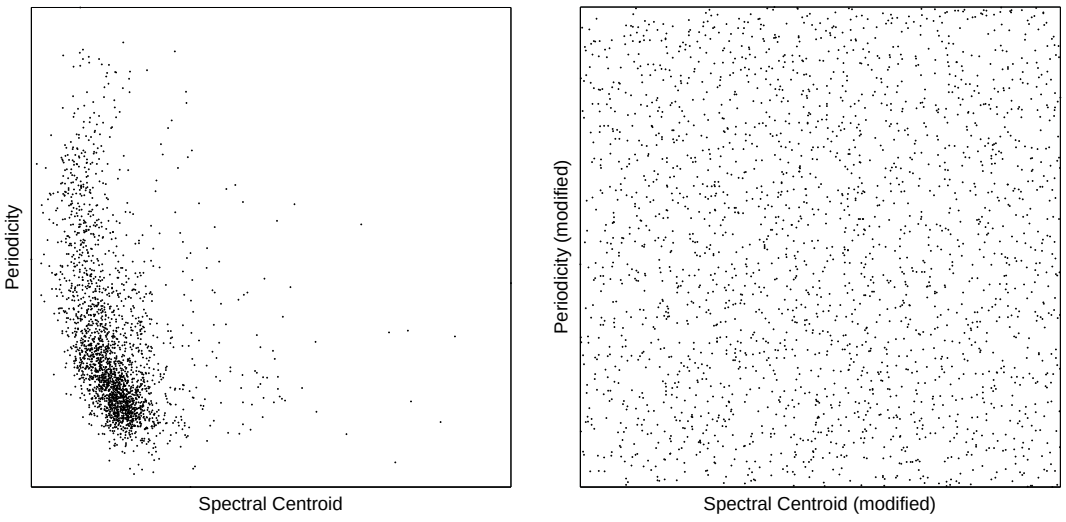

Figure 12: Action of the *domains* algorithm on Spectral Centroid and Periodicity descriptor values (*madeleine* corpus). Left: before transformation; right: after transformation.

4.2 The *grid* algorithm

The *grid* algorithm was developed in order to provide a more conceptually satisfying version of the *domains* algorithm. As no dimension had to be favored over another, the idea of computing the displacement field on a regular grid to interpolate the units displacements was used. This approach extends the concept of dividing the y-axis into smaller areas (introduced in the *domains* algorithm) to both descriptors. The following steps are used:

- **step 1:** the 2-dimensional units density is estimated using the *KDE2D* Matlab toolbox, and the distribution function is computed (cf. Fig. 13);
- **step 2:** the mesh (square grid) on which the density and distribution functions are estimated provide the sampling points on which the displacement field is computed (in this report, 2^8 points were used). On each horizontal or vertical line of the grid, which is made of equally spaced grid points, the corresponding horizontal or vertical section of the 2D distribution function is used to compute the new positions of the grid points according the *uniform* algorithm[7]. On each line, the new positions values are normalised between 0 and 1;
- **step 3:** the new grid provides the coordinates of the new positions of the original grid points. These two grids define the sampled displacement field of the transformation and are used to interpolate the new positions of the units.

Figure 14 shows the action of the algorithm on Gaussian descriptor values. The curved shape in the top right part of the enclosing square can be explained by looking at the distribution function in figure 13. As the origin of the integral that gives the distribution function is located in the lower-left corner of the square, the mesh points located around the top-right Gaussian center only correspond to high distribution function values: as these points are the ones that will be used to interpolate this Gaussian units new positions, this explain why the top-right Gaussian cannot spread across the whole square and is pushed in its upper-right corner. Because of the normalisation of the grid points new positions, one may expect the bottom-left Gaussian center to be fully spread across the square. Displaying the units displacements, as done in figure 15, confirm this prediction.

The fact that the top-right Gaussian center is pushed in the upper-right corner of the square, and that it gets mixed with units from the bottom-left Gaussian center (cf. Fig. 15), is clearly a problem here. However, since the algorithm does not favor one dimension over another, it is a good candidate for an iterated algorithm. Figure 16 shows the action of the algorithm on Gaussian descriptor values, after 10 and 20 iterations. The iterated algorithm seems to work pretty well in spreading the units, although one may expect the introduced distortion to be very high in this case. Interestingly, a diagonal line of very close units can be seen in figure 16 (left). As displaying the units displacement for the transformation shows it (cf. Fig. 17), the units on both sides of this line correspond to the two original Gaussian centers, with the exception of a few units from the lower-left Gaussian center that end up in the upper-right side of the line.

[7] Unlike the *domains* algorithm, the mathematical formulation of the *uniform* algorithm is there crucial, as horizontal and vertical displacements are computed according to sections of the 2-dimensional density.

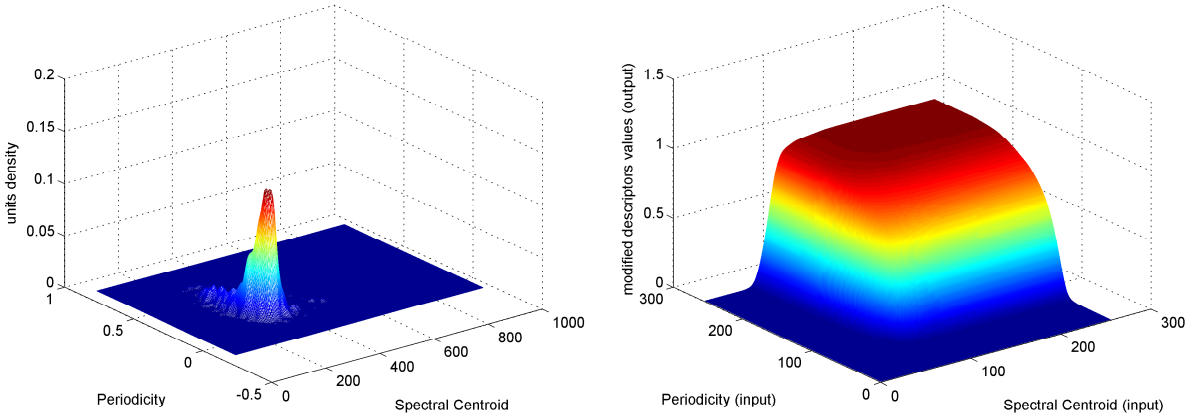

Figure 13: Units 2D density (*madeleine* corpus) (left) and corresponding primitive (right).

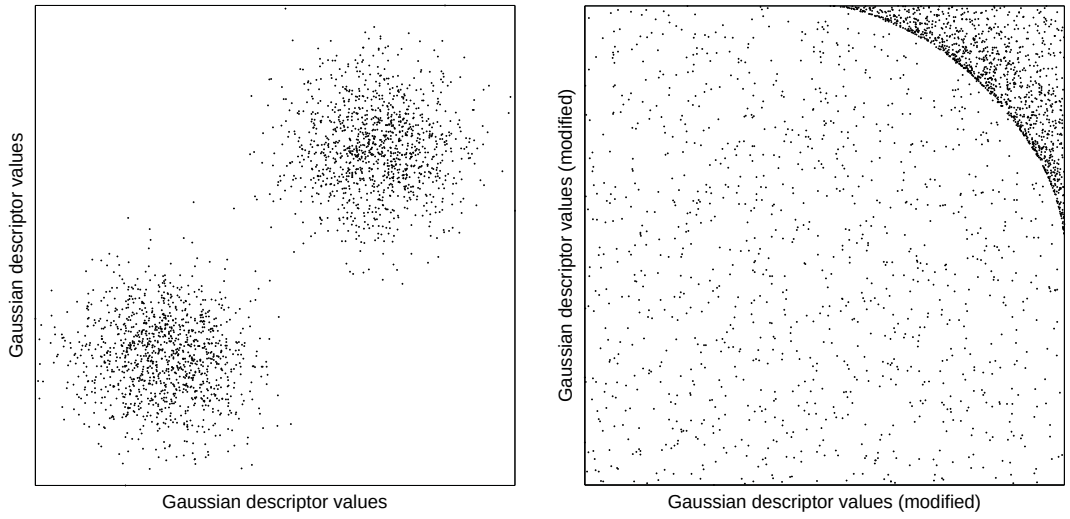

Figure 14: Action of the *grid* algorithm on Gaussian descriptor values. Left: before transformation; right: after transformation.

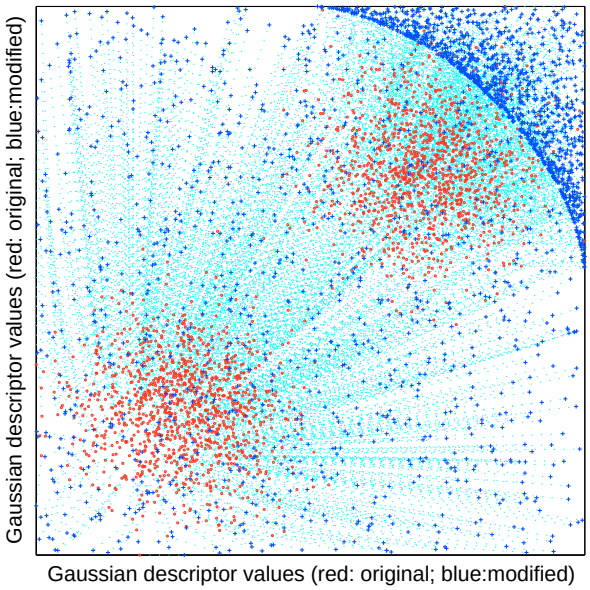

Figure 15: Displacement field (cyan dashed lines) for the *grid* algorithm from original Gaussian units positions (red) to modified units positions (blue).

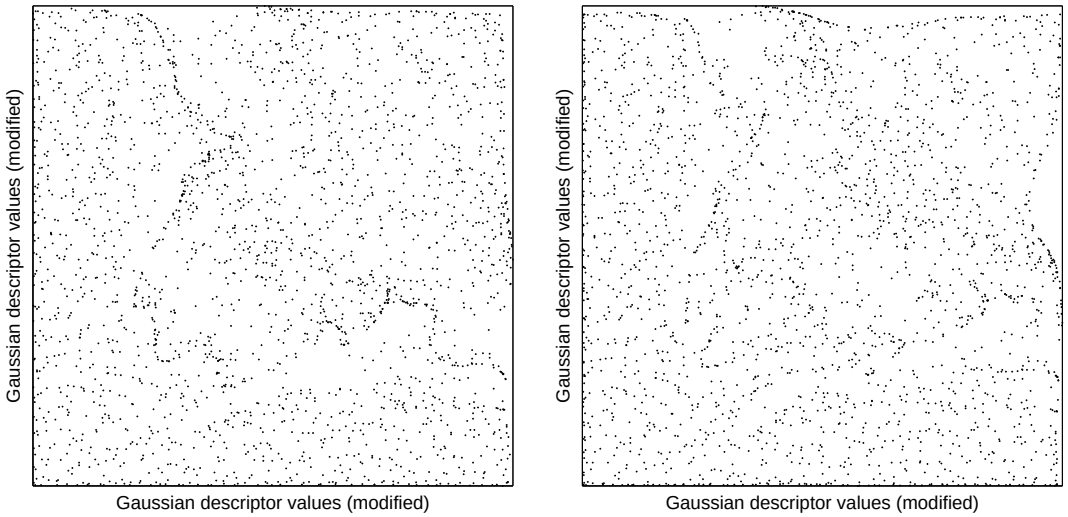

Figure 16: Action of the iterated *grid* algorithm on Gaussian descriptor values. Left: after 10 iterations; right: after 20 iterations.

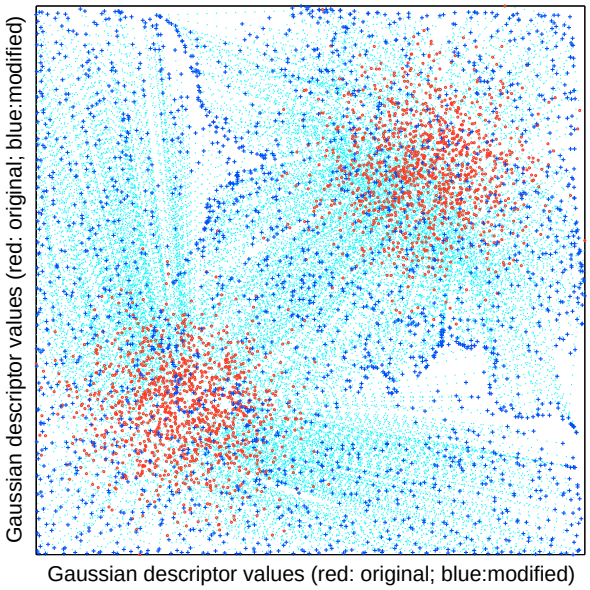

Figure 17: Displacement field (cyan dashed lines) for the iterated *grid* algorithm from original Gaussian units positions (red) to modified units positions (blue) (after 10 iterations).

4.3 A conformal mapping-based algorithm: the *conformal* algorithm

A conformal map refers to a function that preserves angles. Conformal mapping usually takes the form of a mapping between two domains of the complex plane, and is widely used in fluid mechanics to map a complex geometry to a simpler one in order to compute the flow properties. As shown in figure 18, the angle-preservation property ensures that cells of a grid don't interpenetrate each other during the transformation, which implies that the relative positions of the cells in the transformed bounding shape remain the same as in the original grid. For this reason, an algorithm based on conformal mapping could introduce reasonable distortion, besides being conceptually satisfying due to the true 2-dimensional nature of the geometric transformation behind it.

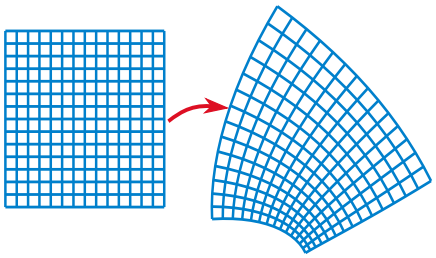

Figure 18: Conformal mapping of a grid.

The *conformal* algorithm is based on a particular class of conformal mappings known as the Schwarz-Christoffel mapping. For any polygon in the complex plane, the Riemann mapping

theorem states that there is a bijective biholomorphic mapping f from the upper half-plane to the interior of the polygon, that maps the real axis to the edges of the polygon. If the polygon vertices are denoted $w_1, \ldots, w_n$, and $\alpha_1\pi, \ldots, \alpha_n\pi$ are the interior angles at the vertices, the mapping f is given by:

$$f(z) = f(z_0) + c \int_{z_0}^{z} \prod_{j=1}^{n-1} (\zeta - z_j)^{\alpha_j - 1} \, \mathrm{d}\zeta$$

were $z_1, \ldots, z_n$ are the real pre-images of the vertices (cf. Fig. 19). They satisfy $z_1 < z_2 < \ldots < z_n = \infty$. The main difficulty with this formula is that in most cases, the prevertices cannot be computed analytically. Although details of the numerical calculations of the Schwarz-Christoffel mapping parameters are beyond the scope of this report, it has to be noted that three of them can be fixed arbitrarily before the other ones can be determined uniquely by solving a system of nonlinear equations. Once this parameter problem is solved, the constants c and $f(z_0)$ are determined uniquely by the polygon [3].

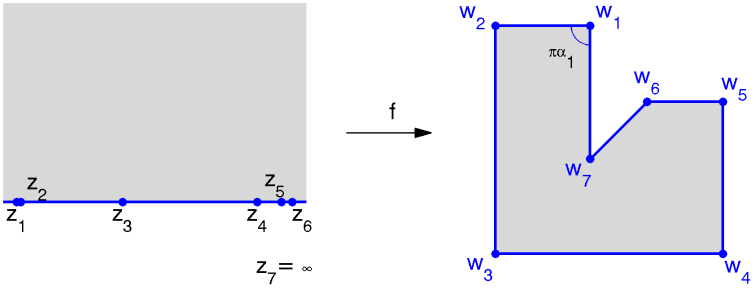

Figure 19: Notational conventions for the Schwarz-Christoffel transformation.

Composing the Schwarz-Christoffel formula with standard conformal maps leads to variations that allow mappings from other domains. In this internship, I focused on mappings from a square and a disk onto a polygon[8]. Although the Schwarz-Christoffel variations give the maps from a disk or a square onto a polygon, which corresponds to the opposite order of what the algorithms studied in this internship are supposed to do, the bijective character of the mappings allow to numerically obtain maps from a polygon to a disk and a square. Consequently, one has first to define a polygon in the corpus projection plane prior to using conformal mapping. Drawing a polygon that encloses all the units would obviously result in a far less efficient transformation than one based on a polygon enclosing only the high-density regions of the projection plane. However, this poses a problem as the mapping is not defined outside the polygon. Consequently, the transformations I realised with conformal mappings deleted the points outside the polygon, which I defined by hand to conduct the first tests. Figure 20 shows units positions for the *madeleine* corpus with Spectral Centroid and Periodicity descriptors, as well as the polygon used to define the Schwarz-Christoffel mapping.

The *conformal* algorithm starts by defining a polygon in the projection plane of the corpus, then transforms the enclosed units according to a Schwarz-Christoffel map. The numerical calculations are carried on by the *Schwarz-Christoffel* toolbox for Matlab [3],[4]. Figure 21

[8] I will continue to refer to these mappings as Schwarz-Christoffel mappings.

shows the action of the *conformal* algorithm on Spectral Centroid and Periodicity descriptor values for the *madeleine* corpus, with mappings to a square and a disk. This algorithm seems less efficient than previous solutions, and poses the problem of choosing how to define the input polygon and the algorithm behavior towards units located outside it. For these reasons, I chose to keep the *conformal* algorithm in a draft state to focus on other solutions. As it causes the deletion of some of the units, this algorithm is not part of the evaluative study conduced in 6.

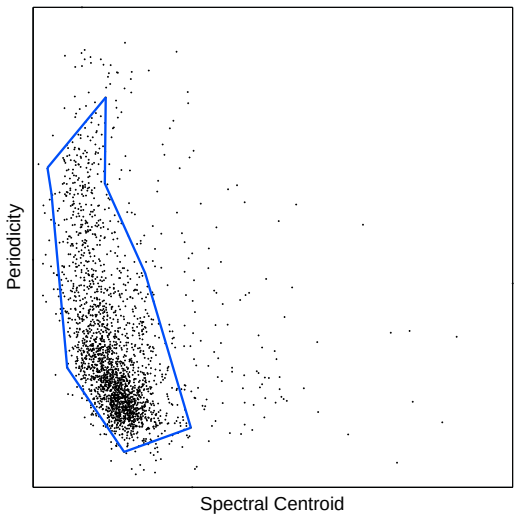

Figure 20: Units positions for *madeleine* corpus with Spectral Centroid and Periodicity descriptors, shown with the polygon used in the *conformal* algorithm.

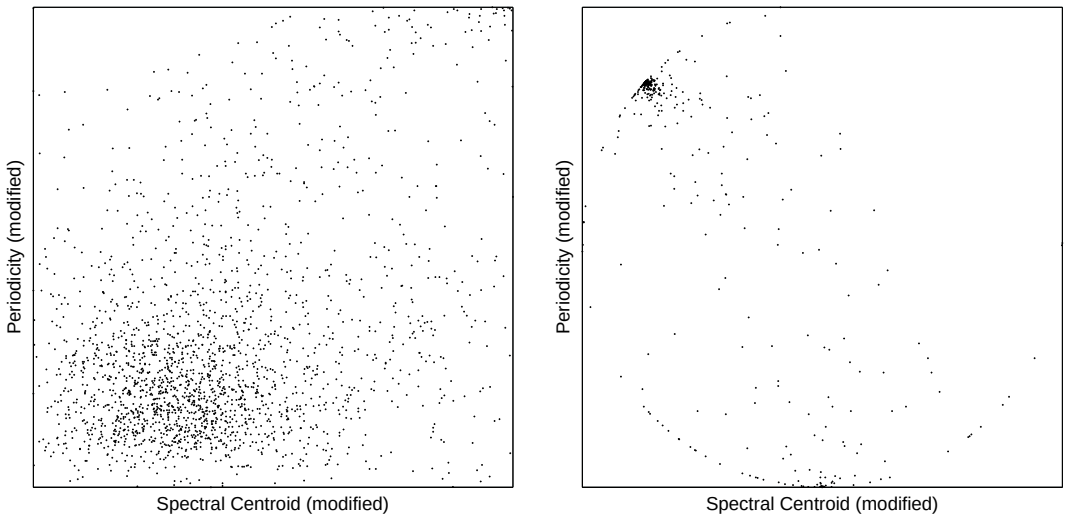

Figure 21: Action of the *conformal* algorithm on Spectral Centroid and Periodicity descriptor values (*madeleine* corpus). Left: mapping to a square; right: mapping to a disk.

5 Physical and hybrid algorithms

5.1 The *distmesh* toolbox for mesh generation

The mesh-based approach used in the *grid* algorithm motivated the search for a way to define a mesh over the units set initial geometry. The idea was to transform this mesh to a regular square grid, and use the old and new meshes to interpolate the units new positions (as in the *grid* algorithm). The *distmesh* Matlab toolbox [9] generates unstructured triangular meshes using a physical algorithm. The authors' motivation was to "develop a mesh generator that [could] be described in a few dozen lines of Matlab" [9]. This simplicity and the physical nature of the algorithm were the essential reasons why I chose to use this toolbox. It was flexible enough to allow me to use it in a different way than the one it was intended to, and develop the *delaunay* algorithm detailed in 5.2.

The *distmesh* algorithm is based on a simple mechanical analogy between a triangular mesh and a 2-D truss structure, or equivalently a structure of springs. The desired mesh geometry can be specified by providing a desired edge length function $h(x, y)$. The region over which the mesh is to be defined 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 0 on its boundary. The algorithm proceeds according to the following steps:

- **step 1:** an initial, uniformly distributed set of points (the future mesh vertices) is created in the bounding box enclosing the region in the form of a mesh of equilateral triangles;
- **step 2:** points outside the region are removed. If the desired mesh is non-uniform, further points are removed according to a probability distribution defined in [9] to improve the algorithm convergence speed, which is higher when the initial points distribution is not uniform[9];
- **enter loop:**
  - **step 3:** a Delaunay triangulation[10] of the points is performed during the first iteration. In further iterations, the triangulation only takes place again if mesh points have moved more than a control distance *ttol* during the previous iteration;
  - **step 4:** mesh points positions are updated according to a physical algorithm. The triangulation 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 has a force-displacement relationship $f(l, l_0)$ depending on its current length l and its unextended length l_0 (which is constant for uniform meshes). A linear spring force is chosen for repulsive forces ($f(l, l_0) = k(l_0 - l)$ for $l < l_0$), and no attractive forces are allowed ($f(l, l_0) = 0$ for $l \geq l_0$). This repulsive-forces only model is designed to help points spread out across the whole geometry, hence generating a proper mesh.
  - **step 5:** points that go outside the region during step 4 are moved back to the closest boundary point. This corresponds to the physical image of external reaction

[9] This feature will not be described in this report, as the *delaunay* algorithm does not make use of it.

[10] A Delaunay triangulation for a set P of points in the plane is a triangulation DT(P) such that no point in P is inside the circumcircle of any triangle in DT(P).

forces on the truss structure, that act normal to the boundary; hence points can move along the boundary, but not go outside.

– **if** all interiors nodes move less than a control distance *dptol*, **exit loop**. Otherwise, go back to **step 3**.

Figure 22 shows the steps leading to the generation of a non-uniform triangular mesh over a region of the plane.

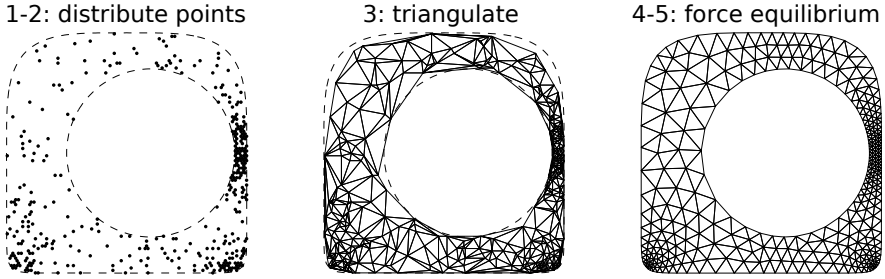

Figure 22: Steps leading to the generation of a non-uniform triangular mesh.

5.2 A *distmesh*-inspired physical algorithm: the *delaunay* algorithm

Studying the way the *distmesh* algorithm worked, I had the idea of applying it to the units in the projection plane, taking them as the initial set of points of the *distmesh* algorithm (the "vertices"). This modified version of the *distmesh* algorithm, where steps 1 and 2 are replaced by loading the units coordinate values (which are first normalised so that no unit is outside the region over which we want them to spread) constitutes the *delaunay* algorithm, which only makes use of the uniform-mesh generating capabilities of the *distmesh* algorithm. Although only very partially explored in this internship, the opportunity to generate non-uniform meshes (which translates in this context as non-uniform units distribution over a desired region) is very interesting and could give birth to more configurable algorithms, as described in 6.4. It has to be noted that, if the purpose of the *distmesh* algorithm is to generate a mesh which is often the intermediate step of a bigger process, the output of the *delaunay* algorithm is no longer considered as a mesh since it constitutes the transformed units positions themselves. In other words, although based on the same physical model, the *delaunay* algorithm is used in a very different context than *distmesh*, of which it constitues an unexpected variation.

The action of the *delaunay* algorithm was first observed for a mapping onto a square region, which contained the initial units positions. Due to the physical nature of the algorithm, parameters were expected to have a big influence on whether it would converge or not. However, default parameters of the *distmesh* toolbox gave excellent results in terms of robustness, with no cases of non-converging situations observed. Besides these default values, the signed distance function of the square region had to be provided, which is a fairly simple task since it can be computed analytically. The initial units descriptor values are shifted so that the units distribution barycenter corresponds to the center of the destination square, in order to prevent it to be located too close to the square boundaries (which would slow down the convergence). I found that normalising the descriptor values so that to leave a small blank area between the units and the square boundaries further accelerated the convergence of the algorithm. As

can be seen on figure 23, the algorithm performs very well on Gaussian descriptor values. It produces perfectly equally-spaced units positions, and the displacements from the original Gaussian centers to the new positions appear to be symmetrical, which is satisfying being given the symmetrical nature of the initial distribution.

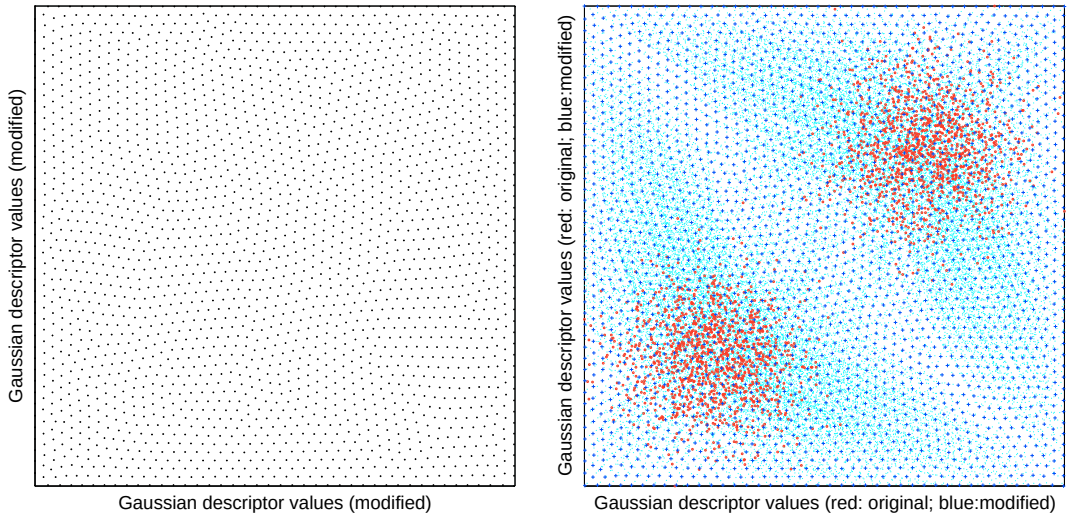

Figure 23: Left: action of the *delaunay* algorithm on Gaussian descriptor values (mapping to a square); right: corresponding displacement field (cyan dashed lines) from original units positions (red) to modified units positions (blue).

Figure 24 shows the action of the algorithm on Gaussian descriptor values, with mapping to a disk. Mapping to a disk is not more difficult than mapping to a square, since the signed distance function of the circle can be computed analytically. This is clearly a strong advantage of this algorithm, since it could allow the units to be mapped to circular interaction devices such as the ReacTable for instance. As can be seen by plotting the displacements from the original units positions to the new ones, the transformation also seems symmetrical in this case.

In order to allow mapping to more general geometries, which could be useful in the case of sound installations for instance, or in the *Topophonie* project, mappings to polygons were also considered. Although the *distmesh* toolbox provided an algorithm for computing numerically the signed distance function of a polygon, I found that the *delaunay* algorithm was not able to converge using it. Consequently, I looked for another way of estimating the signed distance function of a polygon and used an iterative signed distance function estimator implemented in the Matlab *Toolbox of Level Set Methods*. This routine enables to compute the signed distance function on each node of a regular grid defined over the polygon, which gives a sampled signed distance function estimation. This estimation is then provided to a simple routine of the *distmesh* toolbox which uses it to interpolate the signed distance function value at any point of the plane.

The theory of level set methods, how it relates to signed distance functions estimation and the way it is implemented in the used Matlab toolbox is beyond the scope of this internship report. For this reason, and also because of the prospective nature of the polygonal mappings this estimation was needed for, I chose to invest more time on tuning the parameters of

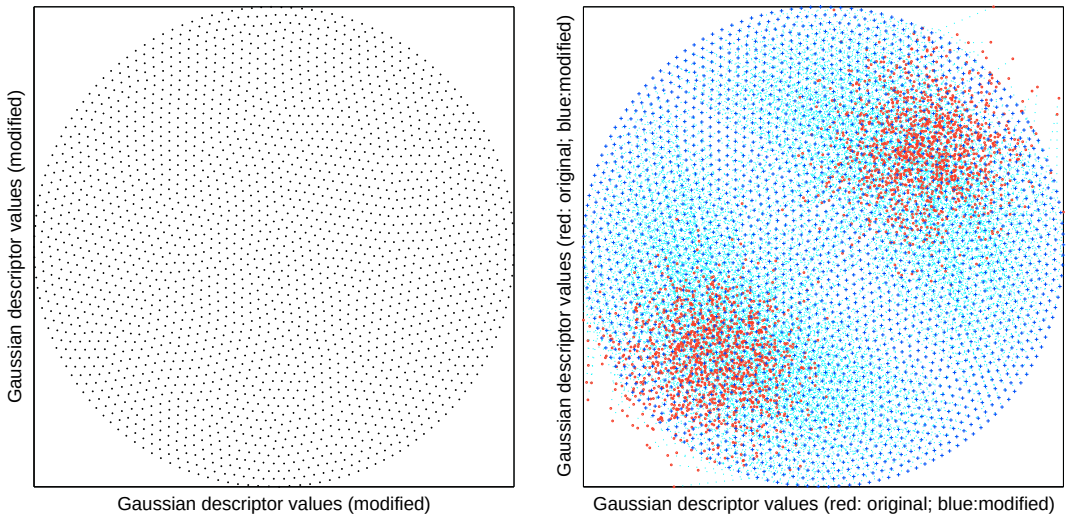

Figure 24: Left: action of the *delaunay* algorithm on Gaussian descriptor values (mapping to a disk); right: corresponding displacement field (cyan dashed lines) from original units positions (red) to modified units positions (blue).

the *delaunay* algorithm to improve the robustness of the algorithm. However, although the *delaunay* algorithm with optimised parameters for mapping to polygons was able to converge in most cases, a few unsuccessful situations remained. Corresponding results are presented in A.2.

5.3 A hybrid algorithm: the *uniform2D/delaunay* algorithm

This improved version of the *delaunay* algorithm is an attempt to adapt the initial units distribution to be as close as possible to the *distmesh* algorithm context, to improve robustness and speed. In the *distmesh* algorithm used to generate a uniform mesh, initial mesh points are uniformly distributed. As units play the role of mesh points in the *delaunay* algorithm, I figured out that the *uniform2D* algorithm could be applied to them before any further transformation, in order to make their distribution closer to the uniform case. Besides improved robustness and convergence speed, lower distortion values were also expected with this approach as the *uniform2D* algorithm introduced no distortion and the first normalisation it produced allowed better positioning of the units in the enclosing shapes onto which the *delaunay* algorithm would map them (square, circle or polygon). Figure 25 describes the *uniform2D/delaunay* algorithm in the case of the mapping to a square. Results and distortion measure values are presented in 6.3.

6 Evaluation

6.1 Quantitative distortion measure

Though visual examination of the algorithms outputs provides a first criterion for evaluating their efficiency, it is necessary to compare them in terms of the amount of distortion they

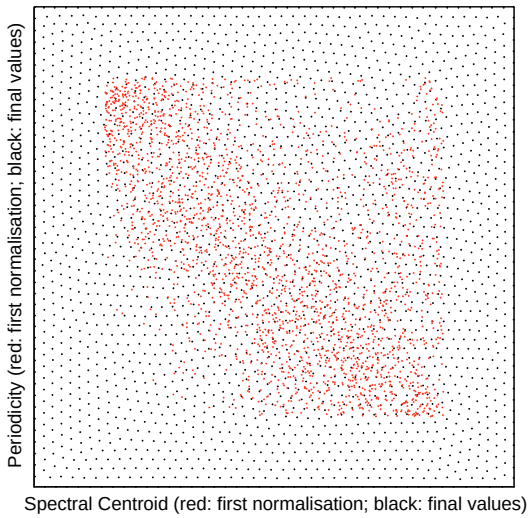

Figure 25: The *uniform2D/delaunay* algorithm (mapping to a square). First step (red): the *uniform2D* algorithm is applied on descriptor values, and resulting values are normalised to fit a smaller square than the one to map to; second step (black): the *delaunay* algorithm is applied on the normalised descriptor values.

introduce during the transformation. To perform this task, I developed an algorithm that provides different informations to assess the introduced distortion. Since the zero-distortion case had been defined as the case where the transformation preserves the order of the units projection on each axis, the algorithm computes measures that help to assess how far a specific algorithm is from this ideal case. We decided that considering unique pairs of units would be the best way to check if one or two of their coordinates had been exchanged during the transformation.

With this data structure in mind, I designed the representation shown in figure 26: one of the units of the pair is located at the center of a Cartesian coordinate system, while the other unit positions relatively to it are plotted after and before the transformation. The four quadrants of the coordinate system correspond to the four main situations where distortion may or may not be introduced during the transformation of these two units. The zero-distortion case corresponds to case 1 shown in figure 26, where B (before transformation) and B (after transformation) are located in the same quadrant. Cases 2 and 4 correspond to the inversion of one of the two descriptor values during the transformation: as B (before transformation) and B (after transformation) are not located in the same quadrant, the relative order of the A and B values of one descriptor has been exchanged. Cases 2 and 4 introduce the same amount of distortion. Case 3 corresponds to the worst-case scenario in terms of introduced distortion, with both descriptor values exchanged during the transformation. The algorithm returns the percentages p_1 and p_2 of units pairs that have exchanged respectively one and two of their coordinates during the transformation.

Although the percentages returned by the algorithm are the most neutral and reliable distortion measure available, it might be useful to dispose of a numerical distortion measure that would sum up the effects of the 4 aforementioned possible cases. This is carried out by attributing a distortion value $d_0 = 0$ to case 1, a single value $d_1 > 0$ to cases 2 and 4, and a value $d_2 > d_1$ to case 3. Because abrupt transitions between these values could cause a small

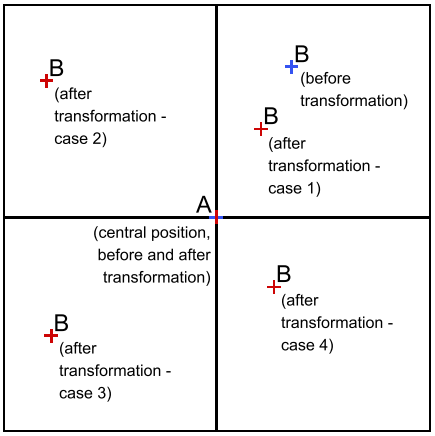

Figure 26: The different cases used in computing the distortion measure du to relative position changes.

angular displacement of B to result in a significant increase of the distortion measure (in the cases where unit B is located close to one of the Cartesian axes of the A-centered coordinate system), a smoothed curved is used. Figure 27 shows the distortion value attributed to a pair of units in function of B (after transformation) angle in the A-centered coordinate system, in the case where B is initially located in the first quadrant. Shifted functions (by multiples of $\frac{\pi}{2}$) are used in cases where B (before transformation) is located in another quadrant. The computation is carried out for each unit pair and the mean over all pairs gives the distortion measure d for the transformation. In this report, the following values were used for this part of the algorithm: $d_1 = 0$ and $d_2 = 10$, so that d is located between 0 (no distortion) and 10 (highest possible distortion).

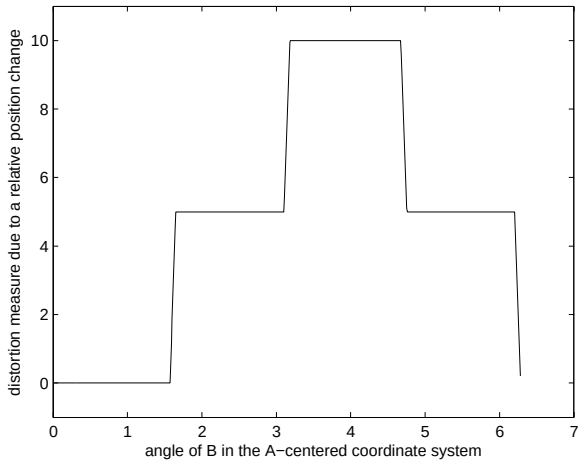

Figure 27: Distortion measure due to a relative position change, when B (before transformation) is in the first quadrant (cf. Fig. 26).

6.2 Evaluation: selected corpora

I chose to evaluate the algorithms on five selected corpora, which correspond to standard geometric configurations as well as cases of application of the CataRT synthesis method.

- the *madeleine* corpus (2404 units), with Spectral Centroid and Periodicity descriptors, is made of environmental sounds and presents a single-centered unit distribution;
- the *gaussian* corpus (2404 units), with 2-centered Gaussian descriptor values on each axis, corresponds to a critical case with two very distinct centers (2404 units);
- the *wave* corpus (2404 units), with Start Time and Periodicity descriptors, is made of environmental sounds and presents no identifiable distribution center (2404 units);
- the *partita* corpus (388 units), with Spectral Centroid and Note Number descriptors, is made of instrumental sounds and presents initial descriptor values ranges dominated by the descriptor values of a few marginal units;
- the *alarm* corpus (779 units), with Spectral Centroid and Note Number descriptors, is obtained using Yin note segmentation on different alarm sounds. Units associated with the same MIDI notes appear as horizontal lines in the 2D display.

Figure 28 shows the initial units distributions of these corpora with corresponding descriptors.

In section 6.3, the *uniform2D*, *domains* (with 10 domains), iterated *grid* (10 iterations), *delaunay* and *uniform2D/delaunay* algorithms are applied on the corpora described in 6.2. As the only mapping available to all algorithms, the mapping to a square is chosen as the main evaluation case. Mappings to a disk and a polygon are presented in A.1 and A.2. Graphical results are presented for all corpora in sections 6.3.1 to 6.3.5, and commented in 6.3.6. Distortion measures are given in 6.3.6 for all algorithms and corpora.

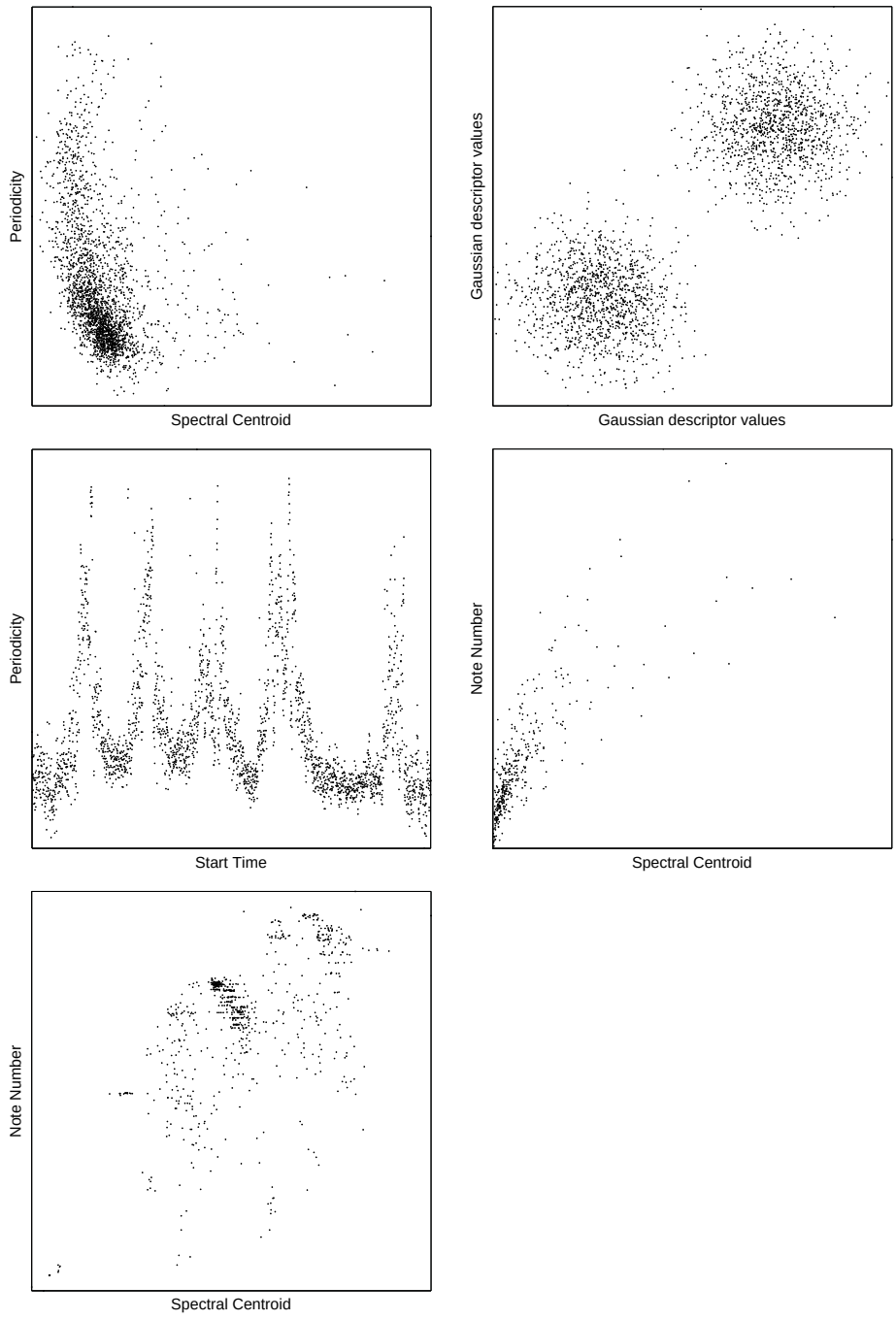

Figure 28: From left to right, top to bottom : the *madeleine* corpus, with Spectral Centroid and Periodicity descriptors; the *gaussian* corpus, with Gaussian descriptor values; the *wave* corpus, with Start Time and Periodicity descriptors; the *partita* corpus, with Spectral Centroid and Note Number descriptors; the *alarm* corpus, with Spectral Centroid and Note Number descriptors.

6.3 Results: mapping to a square

6.3.1 The *madeleine* corpus

Figure 29 shows the results obtained with the *madeleine* corpus.

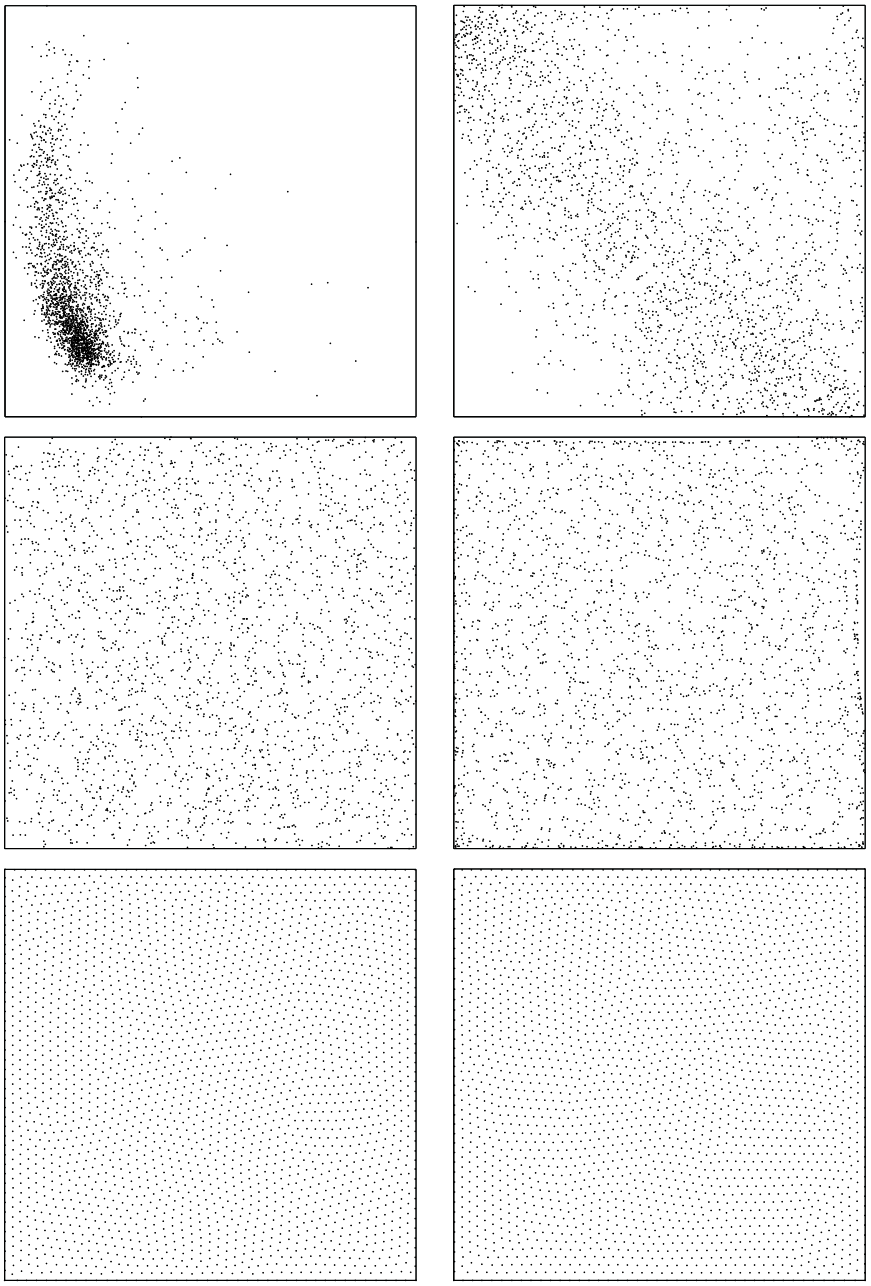

Figure 29: Original and modified *madeleine* corpora (top left: original, then from left to right, top to bottom: *uniform2D*, *domains*, iterated *grid* (10 iterations), *delaunay* and *uniform2D/delaunay* algorithms).

6.3.2 The *gaussian* corpus

Figure 30 shows the results obtained with the *gaussian* corpus.

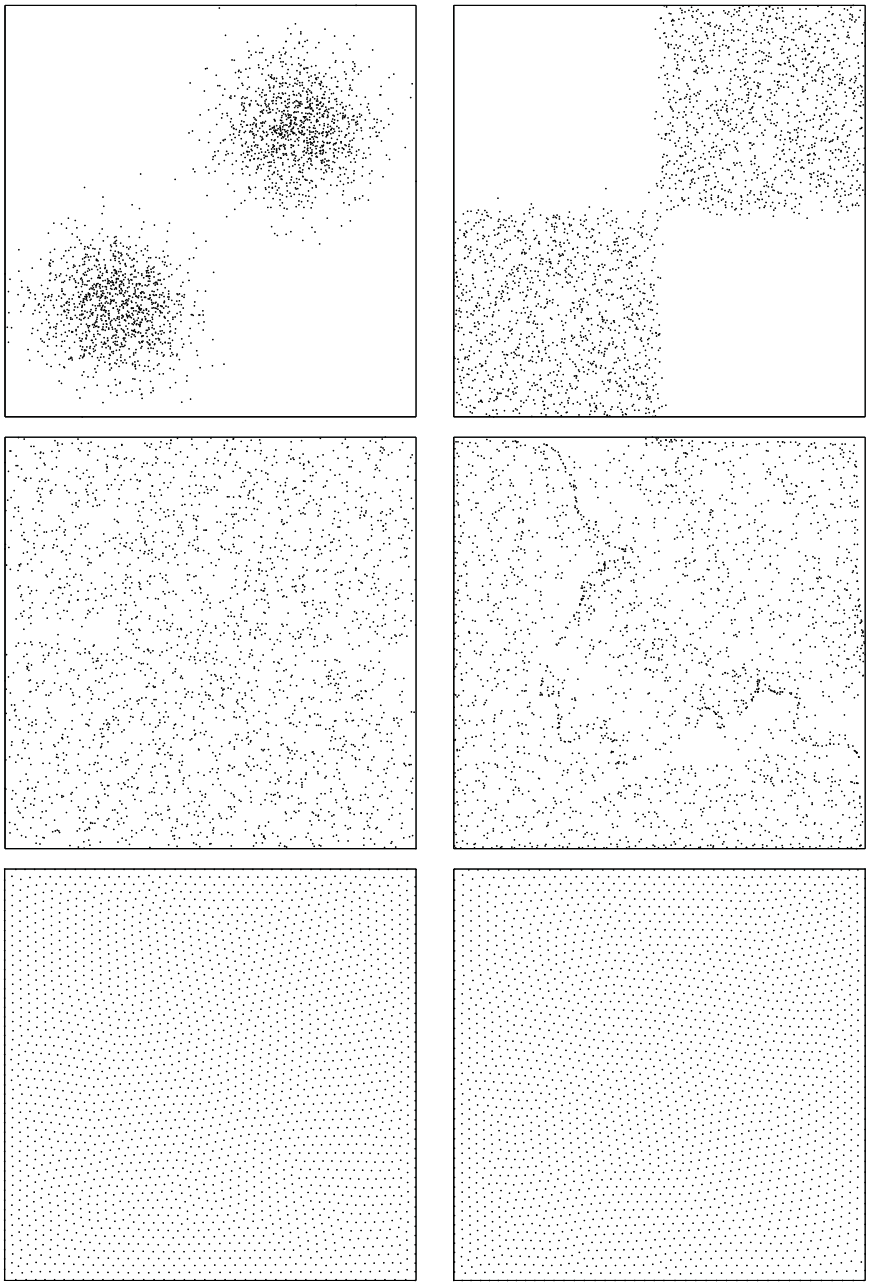

Figure 30: Original and modified *gaussian* corpora (top left: original, then from left to right, top to bottom: *uniform2D*, *domains*, iterated *grid* (10 iterations), *delaunay* and *uniform2D/delaunay* algorithms).

6.3.3 The *wave* corpus

Figure 31 shows the results obtained with the *wave* corpus.

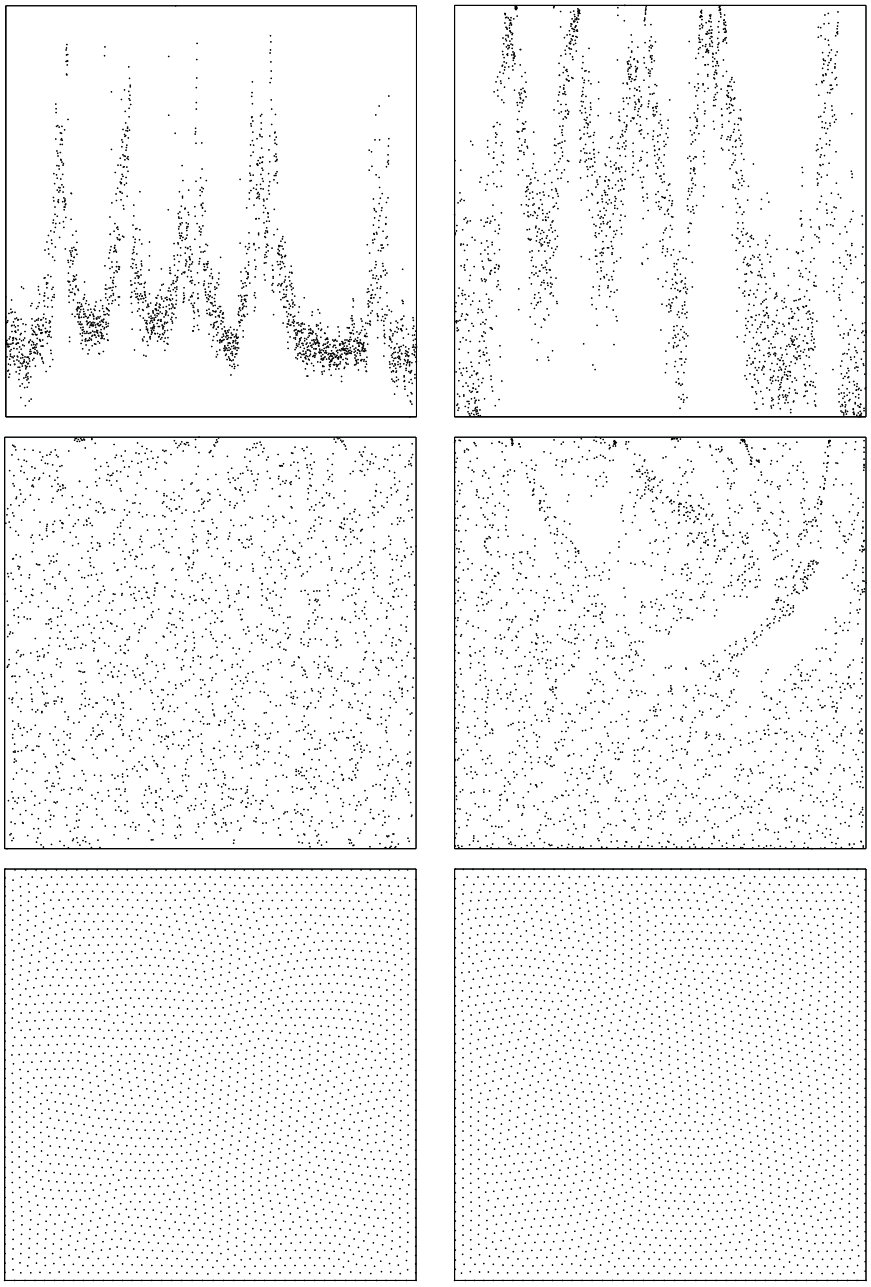

Figure 31: Original and modified *wave* corpora (top left: original, then from left to right, top to bottom: *uniform2D*, *domains*, iterated *grid* (10 iterations), *delaunay* and *uniform2D/delaunay* algorithms).

6.3.4 The *partita* corpus

Figure 32 shows the results obtained with the *partita* corpus.

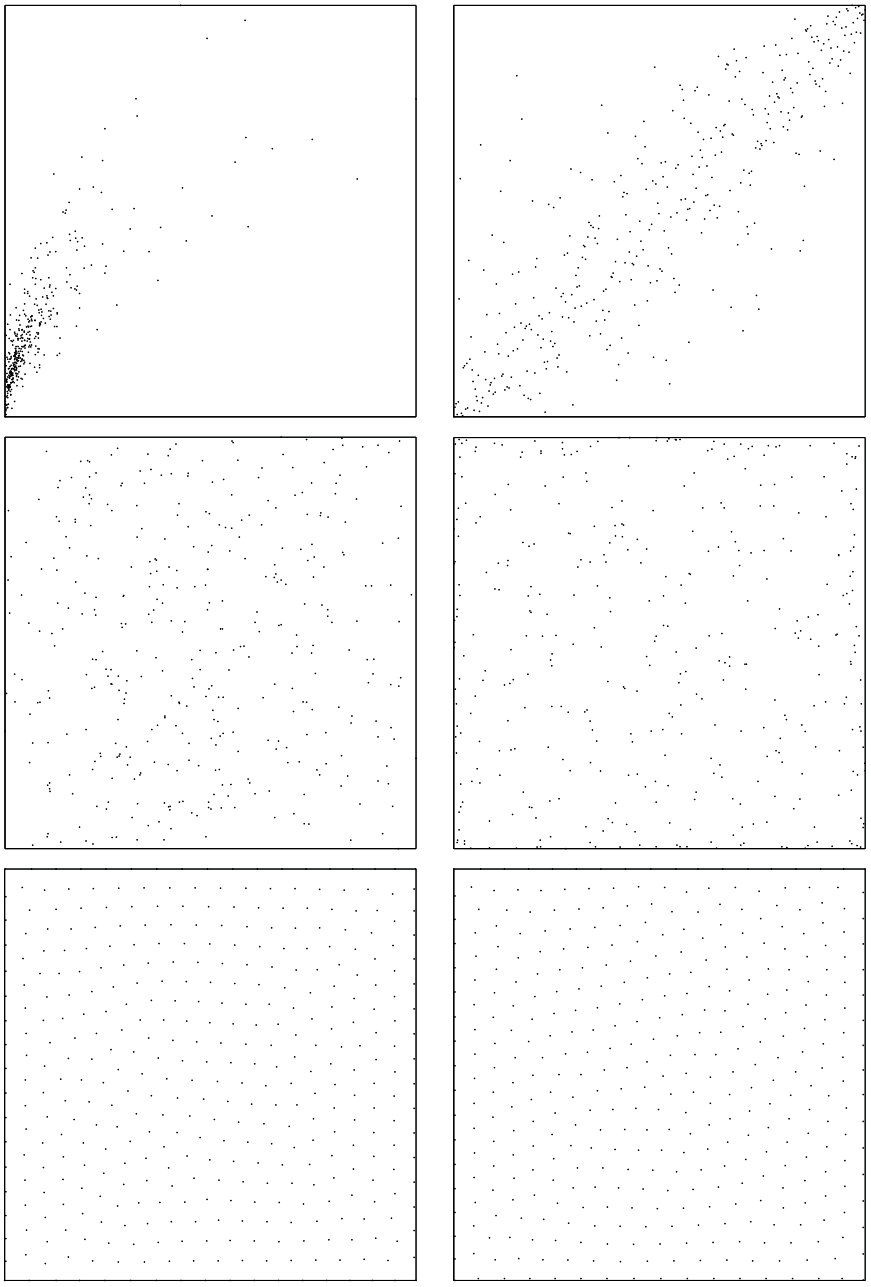

Figure 32: Original and modified *partita* corpora (top left: original, then from left to right, top to bottom: *uniform2D*, *domains*, iterated *grid* (10 iterations), *delaunay* and *uniform2D/delaunay* algorithms).

6.3.5 The *alarm* corpus

Figure 33 shows the results obtained with the *alarm* corpus.

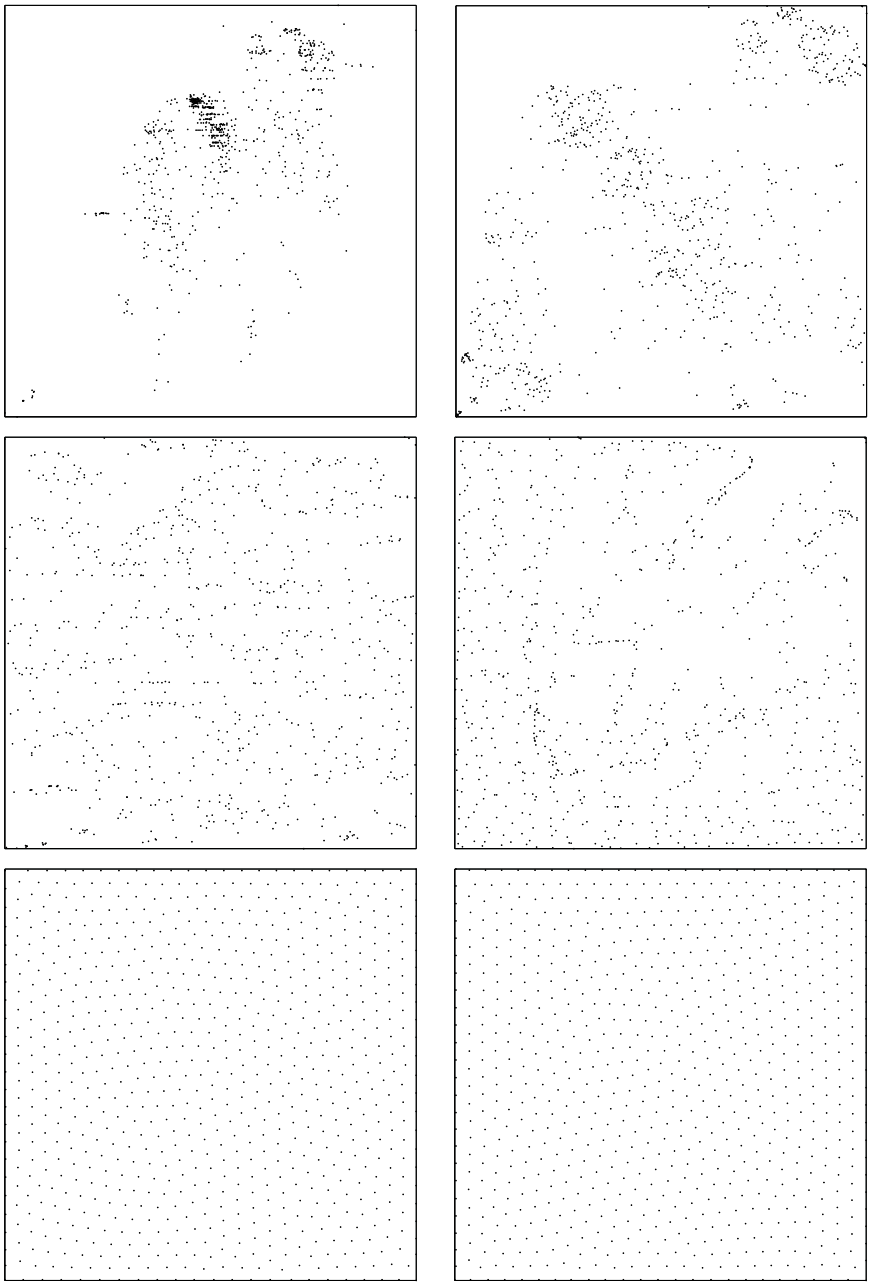

Figure 33: Modified *alarm* corpora (algorithms from left to right, top to bottom: *uniform2D*, *domains*, iterated *grid* (10 iterations), *delaunay* and *uniform2D/delaunay*).

6.3.6 Distortion measures

Distortion measure results are given in table 1. Quantitative distortion measure results (d) are plotted in figure 34.

		madeleine	*gaussian*	*wave*	*partita*	*alarm*
uniform2D	p_1	0 %	0 %	0 %	0 %	0 %
	p_2	0 %	0 %	0 %	0 %	0 %
	d	0	0	0	0	0
domains	p_1	18.3717 %	25.3668 %	11.4501 %	32.1962 %	22.8177 %
	p_2	0.00010391 %	0.0019742 %	0 %	0.046739 %	0 %
	d	0.85656	1.2042	0.5096	1.5401	1.0746
grid (10 iterations)	p_1	19.4499 %	32.79 %	23.5864 %	36.1864 %	34.5124 %
	p_2	0.00034636 %	4.7123 %	0.20837 %	0.014689 %	2.8978 %
	d	0.47009	2.001	0.9911	1.6446	1.5526
delaunay	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	2.9814	1.8637	1.5045	2.7009	1.5806
uniform2D/ delaunay	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	1.0152	1.4087	0.76238	1.5723	0.90526

Table 1: Distortion measures obtained with selected algorithms on evaluation corpora (p_1 and p_2: percentages of units pairs that have exchanged respectively one and two of their coordinates during the transformation; d: quantitative distortion measure).

As expected, the *uniform2D* algorithm introduces no distortion. However, this zero-distortion action prevents the algorithm from spreading efficiently the units across the square, the worst situation being obtained with the *gaussian* corpus (cf. Fig. 29 to 33).

Other algorithms introduce distortion in different amounts. Since the values of the distortion measures depend strongly on the units set initial geometry, algorithms have to be compared based on their action on a single corpus. Reviewing the results corpus by corpus we see that the *domains* algorithm is usually associated with the lowest non-zero distortion values. This could be expected because of the similarities between the *domains* and *uniform2D* algorithms, as well as the interpolation that takes place in the domains algorithm to limit the introduced distortion. Looking at the graphical results in figures 29 to 33, we see that this algorithm performs quite well, though far from producing distributions where all units are equally-spaced (such as distributions obtained with the *delaunay* and *uniform2D/delaunay* algorithms).

The *grid* algorithm, though conceptually more satisfying than the *domains* algorithm (as no dimension is favored over another), seems far less interesting than this one: with a few iterations needed to efficiently spread the units across the square, it takes more time to complete than the *domains* algorithm, and provides less interesting results (figure 16 shows that increasing the number of iterations does not produce more satisfying results). Moreover, the distortion values associated with this algorithm are quite high, because of its iterative application.

Since they both provide the same perfectly uniform graphical results (cf. Fig 29 to 33),

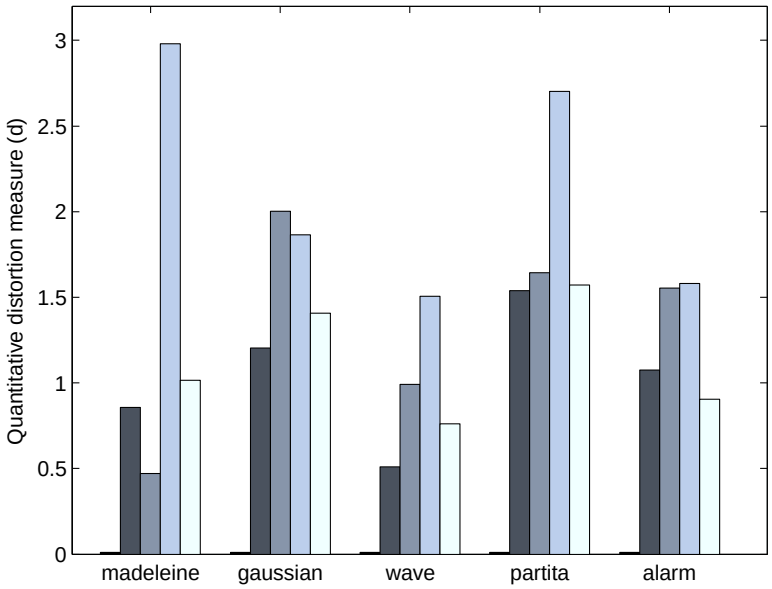

Figure 34: Quantitative distortion measure results obtained with selected algorithms (inside each corpus bar group, from left to right: *uniform2D*, *domains*, iterated *grid* (10 iterations), *delaunay* and *uniform2D/delaunay*).

the *uniform2D/delaunay* algorithm improves the *delaunay* algorithm in every aspect. Although the *delaunay* algorithm is usually associated with the highest introduced distorsion, the distortion measure gives low values with the *uniform2D/algorithm* (comparable with those corresponding to the *domains* algorithm). The fact that applying the *uniform2D* algorithm prior to the *delaunay* algorithm lowers the distortion value can be explained by the fact that this first transformation prevents the points from being too close to each other at the beginning of the *delaunay* algorithm, which would result in anarchic movements of the units caused by repulsive forces. In this context, the 2D-truss structure and the retriangulation process inherited from the *distmesh* algorithm, by ensuring units stay bound to their close neighbors, are responsible for the low distortion values associated with *uniform2D/delaunay*. Besides, the *uniform2D/delaunay* algorithm is faster and more robust than *delaunay*. As the *uniform2D* algorithm (first part of *uniform2D/delaunay*) is much faster than the *delaunay* algorithm (second part of *uniform2D/delaunay*), the *delaunay* and *uniform2D/delaunay* algorithm speeds can be compared by the number of iterations needed for convergence of the physical model they both use. Using the criteria, the *uniform2D/delaunay* algorithm proves to be 1.2 times faster on average than *delaunay*. By taking only into account the number of retriangulations occuring during the iteration (cf. **step 3** of the *distmesh* algorithm), the *uniform2D/delaunay* algorithm appears to be twice faster on average than *delaunay*.

If the *uniform2D/delaunay* algorithm clearly stands out in terms of graphical results, the *domains* algorithm remains an attractive and easier to implement solution. As both algorithms are associated to relatively low and comparable distortion values (although the *domains* algorithm seems to introduce a slightly smaller amount of distortion), the only advantage of *domains* over *delaunay* might be its simplicity, which allows to much faster execution. How-

ever, the *delaunay* algorithm is much more conceptually satisfying, and is able to provide mappings to a disk and any polygon, as can be seen in A.1 and A.2.

6.4 Future improvements

As the most satisfying and configurable algorithm, the *uniform2D/delaunay* is the way by which further improvements on the chosen approach in this internship may be introduced. Particularly, the possibility to define non-uniform edge length function $h(x, y)$ might allow to take into account more than two descriptor values in the output distribution of the units. One can imagine for instance a case where units are still represented in a projection plane defined by two preselected descriptors, but where the distance between units in the transformed distribution is modulated by their similarity in the high-dimensional descriptor space, measured in terms of their euclidean distance. The physical nature of the algorithm allows to introduce this modulation in a very flexible and configurable way, although preliminary tests conduced at the end of the internship have shown that the modified algorithm will encounter convergence issues unless the proper physical model parameters are used.

Although efficient in most cases, the version of the *uniform2D/delaunay* algorithm used for mapping to a polygon still encounters convergence issues, that should be fixed before using it systematically.

The physical nature of the *uniform2D/delaunay* algorithm makes it possible to extend it to three-dimensional point sets. Besides taking an extra descriptor into account, this approach could be useful for applications such as the *Topophonie* project and music interaction (where navigation in a three-dimensional spherical space could be performed by the arm movements of dancers, for instance).

As versions of the *uniform2D/delaunay* algorithm providing mappings to a square and a disk posed no convergence issue and gave excellent results, these algorithms may be integrated to the CataRT software. The implementation may provide further improvements in terms of speed, as the Matlab versions of these algorithms take some time to complete their task, particularly in the case of large corpora.

The chosen approach for this internship supposed that dimensionality reduction had already been performed on the high-dimensional descriptor space (by pre-selecting two descriptors and the corresponding projection plane). However, it seems important to find proper dimensionality reductions algorithms that suit well the CataRT context, in order to determine more precisely how they would compare to the results presented in this report. Particularly, it seems promising to combine a proper dimensionality reduction algorithm with the *uniform2D/delaunay* algorithm.

Finally, as spreading the units in the interaction space might make specific tasks easier (such as finding a particular unit that could have been previously hidden due to overlapping), perceptive tests might be performed to study the advantages of using the solutions developed during this internship in well-defined contexts. Besides, the perceptive effects of the introduced distortion should also be studied, particularly since they were developed with the idea of preserving the physical meaning of the coordinate system axes in mind.

Conclusion

Amongst the algorithms developed during this internship, two solutions seem to address efficiently the problem of spreading the units in a selected enclosing space. The *domains* algorithm is a simple, low-distortion introducing algorithm that performs fast and provide reasonable uniformisation of the initial distribution, though not as good as the *uniform2D/delaunay* algorithm. This last one, however, produces perfectly equally-spaced units distribution with amounts of distortion comparable to those introduced by the *domains* algorithm, though slightly higher. Its physical model nature is also more conceptually satisfying, as it makes it a true 2-dimensional algorithm which can be extensively modified for further improvements.

Both algorithms provide mapping solution to a square, the main encountered interaction spaces in applications. However, the *uniform2D/delaunay* algorithm provides mapping solutions to a disk and any polygon, which extends the range of its possible uses to performances in complex geometrical contexts and *Topophonie* applications.

Further improvements of theses results could notably include taking into account more descriptors from the high-dimensional descriptor space (either by dimensionality reduction algorithms or by introducing the high-dimensional distance as a modulating factor in the *uniform2D/delaunay* algorithm), or taking into considerations the results of tests investigating how the introduced distortion is perceived. However, the square and disk mapping versions of the *uniform2D/delaunay* algorithm can already be used to perform visualisation optimisation tasks, and their implementation in the CataRT software is currently being considered.

References

[1] Thorsten Büring, Jens Gerken, and Harald Reiterer. User interaction with scatterplots on small screens, a comparative evaluation of geometric-semantic zoom and fisheye distortion. *IEEE Transactions on Visualization and Computer Graphics, Vol*, 12, 2006.

[2] Michael Casey. Acoustic lexemes for organizing internet audio. *Contemporary Music Review*, 24(6):489–508, December 2005.

[3] Tobin A. Driscoll. Algorithm 756: A matlab toolbox for schwarz-christoffel mapping. *ACM Trans. Math. Software*, pages 168–186, 1996.

[4] Tobin A. Driscoll. Schwarz-christoffel toolbox user's guide, 2001.

[5] Carl Frederick and Eric L. Schwartz. Vision: Conformal image warping. *IEEE Computer Graphics and Applications*, 10:54–61, 1990.

[6] M. T. Gastner and M. E. J. Newman. Diffusion-based method for producing density equalizing maps. Technical Report physics/0401102, Jan 2004.

[7] Ian M. Mitchell. A toolbox of level set methods version 1.0, 2004.

[8] Ian M. Mitchell. The flexible, extensible and efficient toolbox of level set methods. *J. Sci. Comput.*, 35(2-3):300–329, 2008.

[9] Per olof Persson and Gilbert Strang. A simple mesh generator in matlab. *SIAM Review*, 46:2004, 2004.

[10] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the Cuidado project. Technical Report version 1.0, Ircam – Centre Pompidou, Paris, France, April 2004.

[11] Diemo Schwarz. *Data-Driven Concatenative Sound Synthesis*. Thèse de doctorat, Université Paris 6 – Pierre et Marie Curie, Paris, 2004.

[12] Diemo Schwarz. Concatenative sound synthesis: The early years. 35(1):3–22, March 2006. Special Issue on Audio Mosaicing.

[13] Diemo Schwarz. Corpus-based concatenative synthesis. *IEEE Signal Processing Magazine*, 24(2):92–104, March 2007. Special Section: Signal Processing for Sound Synthesis.

[14] Diemo Schwarz, Grégory Beller, Bruno Verbrugghe, and Sam Britton. Real-Time Corpus-Based Concatenative Synthesis with CataRT. pages 279–282, Montreal, Canada, September 2006.

[15] Diemo Schwarz, Sam Britton, Roland Cahen, and Thomas Goepfer. Musical applications of real-time corpus-based concatenative synthesis. Copenhagen, Denmark, August 2007.

[16] Diemo Schwarz, Roland Cahen, and Sam Britton. Principles and applications of interactive corpus-based concatenative synthesis. In *Journées d'Informatique Musicale (JIM)*, GMEA, Albi, France, March 2008.

[17] Diemo Schwarz and Norbert Schnell. Sound search by content-based navigation in large databases. Porto, July 2009.

[18] Jonathon Shlens. A tutorial on principal component analysis, December 2005.

[19] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. *Visual Languages, IEEE Symposium on*, 0:336, 1996.

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

[21] Sebastian Stober and Andreas Nürnberger. A multi-focus zoomable interface for multi-facet exploration of music collections. In *Proceedings of 7th International Symposium on Computer Music Modeling and Retrieval (CMMR'10)*, pages 339–354, Malaga, Spain, Jun 2010.

[22] Sebastian Stober and Andreas Nürnberger. MusicGalaxy - an adaptive user-interface for exploratory music retrieval. In *Proceedings of 7th Sound and Music Computing Conference (SMC'10)*, pages 382–389, Barcelona, Spain, Jul 2010.

[23] Lloyd N. Trefethen. Numerical computation of the schwarz-christoffel transformation. Technical report, Stanford, CA, USA, 1979.

[24] Hiromitsu Yamada, Kazuhiko Yamamoto, and Taiichi Saito. A nonlinear normalization method for handprinted kanji character recognition–line density equalization. *Pattern Recognition*, 23(9):1023 – 1029, 1990.

[25] Olga Sorkine Yu-Shuen Wang, Chiew-Lan Tai and Tong-Yee Lee. Optimized scale-and-stretch for image resizing. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH ASIA)*, 27(5), 2008.

A Evaluation: mapping to a disk and a polygon

This section presents the results obtained with the versions of the *delaunay* and *uniform2D/delaunay* algorithms providing mappings to a disk and a polygon. Distortion measures are given in A.3.

A.1 Mapping to a disk

Figure 35 shows results obtained with the mapping to a disk version of the *uniform2D/delaunay* algorithm on evaluation corpora. As the *delaunay* algorithm provides graphically equivalent results, they are not displayed.

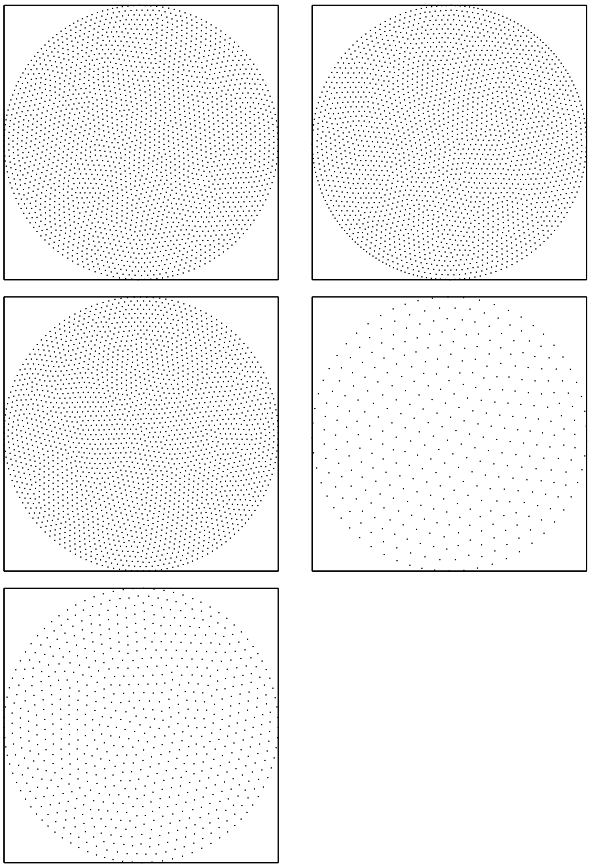

Figure 35: Results obtained with the *uniform2D/delaunay* algorithm on evaluation corpora (from left to right, top to bottom: *madeleine*, *gaussian*, *wave*, *partita* and *alarm*).

A.2 Mapping to a polygon

Mapping results are presented here in the case of three different polygons. Both *delaunay* and *uniform2D/delaunay* algorithms use the same parameters to allow comparing their distortion measures. These parameters have been fine-tuned to ensure convergence of the algorithms in most cases, although further improvements are possible (as demonstrates the general convergence failure in the *partita* corpus case).

First example

Figure 36 shows results obtained with the mapping to a polygon version of the *uniform2D/delaunay* algorithm on evaluation corpora, in the first polygon example case. As the *delaunay* algorithm provides graphically equivalent results, they are not displayed.

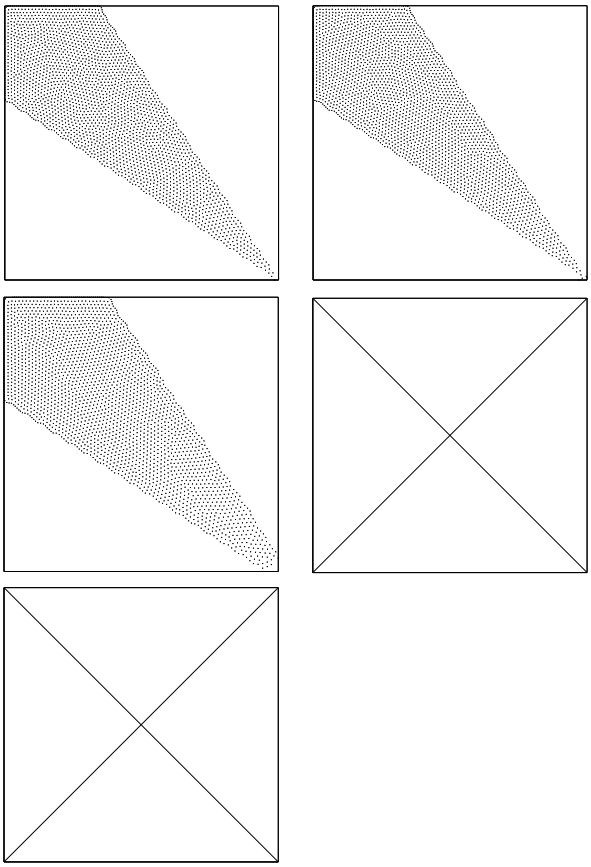

Figure 36: Results obtained with the *uniform2D/delaunay* algorithm on evaluation corpora (from left to right, top to bottom: *madeleine*, *gaussian*, *wave*, *partita* and *alarm*); crossed boxes indicate algorithm does not converge.

It can be noted here that the algorithm stops before fully spreading the units across the whole polygon in the case of the *wave* corpus. Results obtained with the *delaunay* algorithm may display the same kind of artifact, but not necessarily on the same corpora. This illustrates the difficulty to find a set of parameters that would ensure proper convergence of the physical

algorithm in all cases.

Second example

Figure 37 shows results obtained with the mapping to a polygon version of the *uniform2D/delaunay* algorithm on evaluation corpora, in the first polygon example case. As the *delaunay* algorithm provides graphically equivalent results, they are not displayed. However, it has to be noted that the *delaunay* algorithm failed to converge with the *madeleine* corpus in this case.

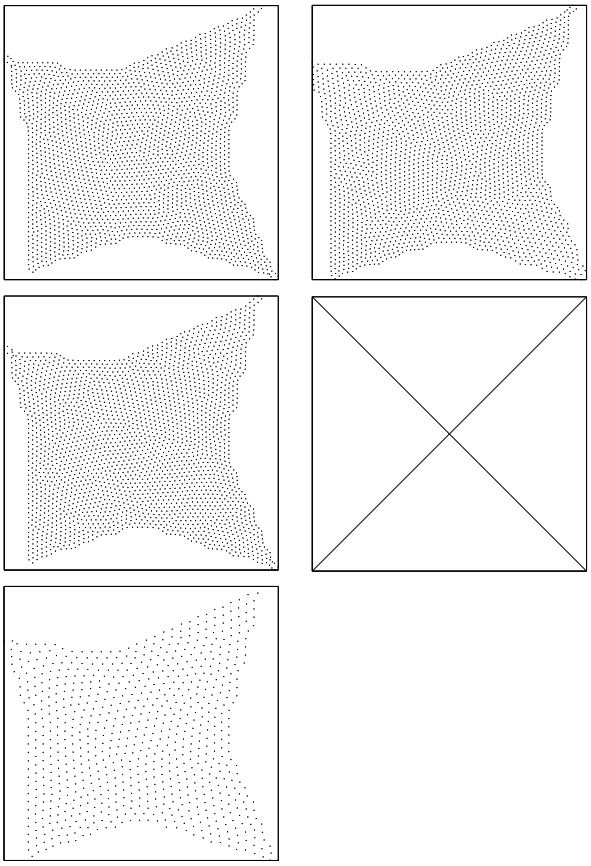

Figure 37: Results obtained with the *uniform2D/delaunay* algorithm on evaluation corpora (from left to right, top to bottom: *madeleine*, *gaussian*, *wave*, *partita* and *alarm*); crossed boxes indicate algorithm does not converge.

Third example

Figure 38 shows results obtained with the mapping to a polygon version of the *uniform2D/delaunay* algorithm on evaluation corpora, in the first polygon example case. As the *delaunay* algorithm provides graphically equivalent results, they are not displayed.

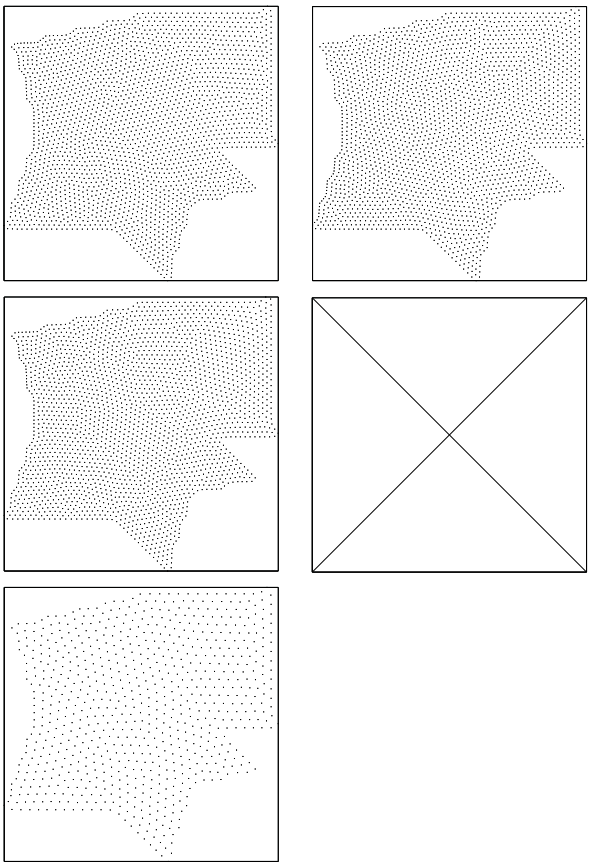

Figure 38: Results obtained with the *uniform2D/delaunay* algorithm on evaluation corpora (shown in the same order as Fig. 28); crossed boxes indicate algorithm does not converge.

A.3 Distortion measures

Distortion measure results obtained with mappings onto a disk and the three polygon examples are given in table 2. Quantitative distortion measure results obtained with mapping onto a disk are plotted in figure 39; results obtained with polygons are not plotted due to missing data (corresponding to cases where one algorithm is not converging).

		madeleine	*gaussian*	*wave*	*partita*	*alarm*
delaunay (disk)	p_1	42.7352 %	33.6393 %	29.0049 %	44.2775 %	29.7087 %
	p_2	9.8733 %	2.7107 %	2.4199 %	6.4847 %	2.909 %
	d	3.0279	1.8373	1.5818	2.7531	1.6586
uniform2D/ delaunay (disk)	p_1	22.5572 %	29.0673 %	17.9267 %	34.3355 %	21.3728 %
	p_2	0.53696 %	0.94198 %	0.4317 %	0.15758 %	0.29837 %
	d	1.0612	1.4272	0.82299	1.6034	0.97591
delaunay (first polygon)	p_1	32.1045 %	55.3688 %	29.0412 %	not converging	not converging
	p_2	8.3129 %	2.8689 %	2.1091 %		
	d	2.3748	2.9777	1.5907		
uniform2D/ delaunay (first polygon)	p_1	19.5114 %	54.9943 %	25.0091 %	not converging	not converging
	p_2	0.83323 %	1.6119 %	0.94212 %		
	d	0.98672	2.8352	1.2659		
delaunay (second polygon)	p_1	not converging	32.5102 %	22.9497 %	not converging	25.2414 %
	p_2		1.7455 %	1.3557 %		2.3024 %
	d		1.6874	1.1734		1.3752
uniform2D/ delaunay (second polygon)	p_1	23.212 %	28.0851 %	15.9882 %	not converging	18.2661 %
	p_2	0.34566 %	0.62268 %	0.27213 %		0.23526 %
	d	1.0797	1.3505	0.7137		0.81595
delaunay (third polygon)	p_1	39.9971 %	32.5102 %	26.4318 %	not converging	24.8872 %
	p_2	5.5671 %	1.7455 %	1.4692 %		2.1431 %
	d	2.4522	1.6874	1.3546		1.3439
uniform2D/ delaunay (third polygon)	p_1	27.0972 %	23.8461 %	20.6493 %	not converging	18.3514 %
	p_2	0.47835 %	0.66175 %	0.42875 %		0.30135 %
	d	1.282	1.1444	0.95967		0.83101

Table 2: Distortion measures obtained when mapping evaluation corpora onto a disk and a polygon (p_1 and p_2: percentages of units pairs that have exchanged respectively one and two of their coordinates during the transformation; d: quantitative distortion measure).

As in 6.3.6, these measures show that in identical contexts, the *uniform2D/delaunay* algorithm introduces lower distortion than the *delaunay* algorithm. Although both algorithms have similar convergence schemes, the *uniform2D/delaunay* algorithms proves to be slightly more robust than *delaunay* by succeeding to converge with the *madeleine* corpus in the case of the first polygon.

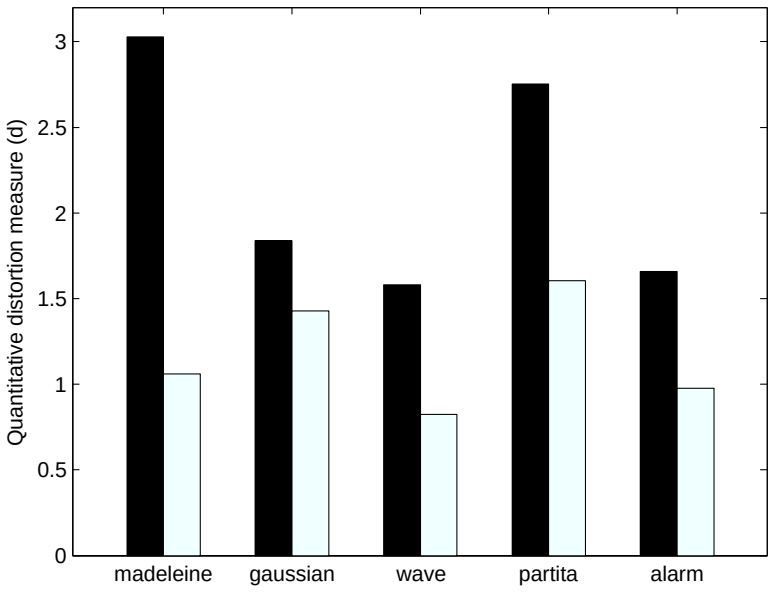

Figure 39: Quantitative distortion measure results obtained with available algorithms for mapping onto a disk (inside each corpus bar group, from left to right: *delaunay* and *uniform2D/delaunay*).