Sound texture synthesis from summary statistics

Hugo Caracalla

Thèse de doctorat
de l'Université Pierre & Marie Curie
École doctorale **Informatique, Télécommunications et Électronique** (Paris)
Sciences & Technologies de la Musique et du Son
(UMR 9912)

Thèse de doctorat d'Informatique

Sound texture synthesis from summary statistics

Par *Hugo Caracalla*

Dirigée par Axel Roebel

Soutenue publiquement le 6 Décembre 2019 devant le jury composé de :

Rapporteurs	Pr. Joshua Reiss	Queen Mary University of London
	Pr. Lonce Wyse	National University of Singapore
Examinateurs	Pr. Carlos Agon	Université Pierre & Marie Curie
	Pr. Elaine Chew	IRCAM
	Dr. Patrick Pérez	valeo.ia
Directeur de thèse	Dr. Axel Roebel	IRCAM

Summary

Sound textures are a wide class of sounds that includes the sound of the rain falling, the hubbub of a crowd and the chirping of flocks of birds. All these sounds present an element of unpredictability which is not commonly sought after in sound synthesis, requiring the use of dedicated algorithms. However, the diverse audio properties of sound textures make the designing of an algorithm able to convincingly recreate varied textures a complex task.

This thesis focuses on parametric sound texture synthesis. In this paradigm, a set of summary statistics are extracted from a target texture and iteratively imposed onto a white noise. If the set of statistics is appropriate, the white noise is modified until it resemble the target, sounding as if it had been recorded moments later.

In a first part, we propose improvements to perceptual-based parametric method. These improvements aim at making its synthesis of sharp and salient events by mainly altering and simplifying its imposition process.

In a second, we adapt a parametric visual texture synthesis method based statistics extracted by a Convolutional Neural Networks (CNN) to work on sound textures. We modify the computation of its statistics to fit the properties of sound signals, alter the architecture of the CNN to best fit audio elements present in sound textures and use a time-frequency representation taking both magnitude and phase into account.

Résumé

Les textures sonores sont une catégorie de sons incluant le bruit de la pluie, le brouhaha d'une foule ou les pépiements d'un groupe d'oiseaux. Tous ces sons contiennent une part d'imprévisibilité qui n'est habituellement pas recherchée en synthèse sonore, et rend ainsi indispensable l'utilisation d'algorithmes dédiés. Cependant, la grande diversité de leurs propriétés complique la création d'un algorithme capable de synthétiser un large panel de textures.

Cette thèse s'intéresse à la synthèse paramétrique de textures sonores. Dans ce paradigme, un ensemble de statistiques sont extraites d'une texture cible et progressivement imposées sur un bruit blanc. Si l'ensemble de statistiques est pertinent, le bruit blanc est alors modifié jusqu'à ressembler à la cible, donnant l'illusion d'avoir été enregistré quelques instants après.

Das un premier temps, nous proposons l'amélioration d'une méthode paramétrique basée sur des statistiques perceptuelles. Cette amélioration vise à améliorer la synthèse d'évènements à forte attaque ou singuliers en modifiant et simplifiant le processus d'imposition.

Dans un second temps, nous adaptons une méthode paramétrique de synthèse de textures visuelles basée sur des statistiques extraites par un réseau de neurones convolutifs (CNN) afin de l'utiliser sur des textures sonores. Nous modifions l'ensemble de statistiques utilisées afin de mieux correspondre aux propriétés des signaux sonores, changeons l'architecture du CNN pour l'adapter aux événements présents dans les textures sonores et utilisons une représentation temps-fréquence prenant en compte à la fois amplitude et phase.

Table of contents

Chapter 1

Introduction

Chapter overview

This thesis revolves around the synthesis of sound textures. This class of sounds may be defined as the random overlapping of myriads of small audio events, hardly perceptible individually, but which may also occasionally contain salient events. From the potential applications of a texture synthesis algorithm, a few objectives may be established: the synthesis needs to be realistic, diverse, flexible, extensible and controllable. An general overview of the thesis is also given at the end of the chapter.

Contents

What are sound textures ?

To be able to discuss the synthesis of sound textures, one first needs to define the class of sounds that they represent. The name evokes their visual counterpart: a wallpaper of sorts, an imbroglio of patches forming a coherent whole. But how does this translate to the audio domain ? The sound the rain makes when falling on a window pane may qualify as a sound texture, as would the howling of the wind and other weather phenomena. The chirps and songs of a flock of bird and the babbling of a crowd in a cafe might also fit the term, although possibly only to a limit: what if there is only a handful of individuals, or one is speaking much louder than the others ? While those questions may seem mostly rhetorical at first, they foretell much of the difficulties encountered in sound texture synthesis, and in our work.

1.1 Defining sound textures

Trying to set the limits of the term "sound texture" and defining it properly is mandatory if we are to work on synthesizing these textures: both to clarify our discussions and to establish guidelines for the algorithm we design. But in this case, moving away from a definition by example is rather arduous.

The question of this definition was first addressed in [Saint-Arnaud 1995], which author initially coined the term "sound texture", and which brought some spotlight on this "large and largely ignored class of audio signal" ([Athineos et al. 2003]). Saint-Arnaud led a series of experiments aimed at gauging the common interpretation people had of this term and establishing a definition of it as precise as possible. The following is largely based on his work, unless stated otherwise.

1.1.1 Properties

When trying to properly define a class of sounds, one way of proceeding is to try and regroup the common characteristics that members of the class possess: from them, one may then outline a set of properties required to be part of said class.

In the case of sound texture, the source (or sources) of the texture could hardly be considered a common factor. While a lot of them originate from environmental sounds (such as water, fire and wind noises), others come from human activity (such as chatting and clapping) or are of mechanical origin (such as engine and traffic noises). On the other hand they all possess a randomness factor, in that it is always hard to describe at a precise level what is happening and when due to the sheer number of events occurring. This unpredictability does however not mean that we are not able to describe a texture, or to judge its properties: while we cannot predict when and how each drop of rain will hit the ground, we can still easily distinguish a drizzle from a downpour simply by listening to it. This would hint at the presence of an overarching organization behind the occurring of events, to which we are sensible.

Plurality and granularity

A common factor of most textures evoked so far is the great number of sources from which each of them originates. One stridulating cricket is ill-qualified to produce a textural background, but there is little doubt that adding a hundred more to the scene would produce a perfectly rural sound texture. In a similar fashion, the sound

of one single car passing by does not generate a sound texture but a busy highway would. This point is more difficult to make for textures such as fire or engine noises, since it could be argued that there technically are not several sources. Despite that, the basic idea stays the same: a lot of events (in those cases, small explosions in the fire or in the engine) each produce a particular sound.

Those sounds act as elementary sonic atoms to the texture: it is only by combining them in great numbers that a sound texture is generated. Examples of atoms are an individual clap, a drop of rain hitting the ground or the singing of one bird. The disparity in complexity between those sounds seems to indicate that it is not crucial that those atoms be as basic as possible, only that they resemble each others and be numerous enough to form a sonic tapestry.

Anonymity and saliency

This profusion contributes to an overall sense of anonymity in textures. The sound of one single woman or man speaking is too understandable to be called a texture (and is rather considered as spoken voice), but the sound of a crowded room where several conversations are taking place and form a sort of hubbub is not. It seems important for the elements of the texture to be indistinct, blurry, so as to blend with each others as much as possible. This in turn contributes to a "background" quality of textures: they fit well as canvas to other sounds (and are often used this way), and as such are often considered bland.

This uneventfulness is characteristic of sound textures and can be assimilated to an absence of saliency from the sources, in the sense that none of them sticks out from the rest. Despite that, it is often possible to track one of the sources by ear (for instance one of the speakers in a crowd), and temporarily discriminate it from the others. Should we consider then that such a sound does not belong to the sound texture class ? We could also imagine a relatively even background (such as a the recording of the rain), in which a source would occasionally emit a very distinctive event (such as thunder): stating whether this sound is or is not a texture would also prove a hard task. This reveals a grey area of sounds containing slightly salient events, testing the limits of our would-be definition.

Randomness and temporal organization

The unpredictability of events seems to be another of the defining features of sound textures. All sonic atoms composing the texture appear to behave at least partly randomly, be it in their properties (e.g. different timbres of voices or size of raindrops) or in their occurring: we would be hard pressed to predict when any of those events is about to happen.

Despite that, textures cannot only be qualified as random. Although the information they carry is relatively weak compared to organized signals such as speech or music, they still tend to inform us on the state of the sources: "how hard is it raining ?" and "how many people are in this room ?" are questions that can be answered with eyes closed. This tends to indicate that although their content is unpredictable at an atomic scale, textures still follow a sort of temporal organization to which we are sensible and from which we may extract information.

1.1.2 Definitions

As is usually the case when trying to put words on such a large and initially vague concept, establishing one precise definition of sound textures is a complex task. This task is made even harder by the fact that this class of sounds is not often talked about, despite being commonly encountered in our daily lives. Deciding whether a sound is or is not a sound texture often comes down to an educated guess, comparing what we hear with the (often vague) mental idea we may have of textures. Luckily, the task at the core of our work is not to come up with a complete definition of them but rather to synthesize them: because of this, we can settle for an incomplete definition.

The discussion over the acceptable saliency of events contained in textures makes clear the fact that there exists several levels of tolerance when deciding what is or is not a texture. Those different interpretations can be simplified and split into two versions of the definition.

Restrictive definition

The first, in line with the work of Saint-Arnaud, is a narrower interpretation that aims at only including sounds that are unanimously perceived as textures. In [Saint-Arnaud and Popat 1995], the following requirements are listed:

- sound textures are formed of basic sound elements, called atoms
- atoms occur following a higher-level pattern (that typically does not span more than a few seconds)
- the pattern must remain constant throughout the texture

Since atoms contained in the texture need to systematically follow a high-order organization, this definition discards sounds including singular events that are not (or only erratically) repeated.

At first glance, this leaves room for uneventful sounds such as rain, radio static and the like. But looking further, it creates another issue: realistically speaking, any recording of a sound texture is likely to contain a recognizable event that will break from this definition. Be it a burning log collapsing in the fire or one bird emitting a characteristic cry, any salient event that is not repeated within seconds automatically falls outside of the higher-level pattern of the texture. Because of those events are outside of the scope of this definition, any sound texture synthesis algorithm built on it risks producing unrealistic, overly synthetic sounds. It is for this reason that we decided to move away from Saint-Arnaud's definition and adapt it into a looser one.

Lenient definition

We understand that it is probably impossible to set clear, "acceptable" limits on a concept such as saliency, and even more so on a concept as subjective as a class of sounds. Instead of engaging in this adventurous task, we add the following points to the previous definition in order to outline a broader definition of sound textures:

- in addition to atoms, singular events may be present in the texture
- those events must be occasional compared to the time scale of the pattern

- no strong information should be conveyed by them

This definition is loose enough in order to include all examples mentioned up to this point, and fits a more realistic vision of sound textures. The condition regarding the information conveyed is added to ensure that while salient events may be present in it, the texture does not break away from the idea of an anonymous background carrying only little information. This way our definition still excludes music and understandable speech, and fits the general idea we have of sound textures.

Again, this definition is not final and we do not hold the answer to the nature of sound textures: the search for the limits of the term is meant as a way for us to set the ground for sound texture synthesis, and as a reflection on which to base our research.

1.2 Motivation

As the title of this thesis may have given away, the goal at its core is the creation of a sound texture synthesis algorithm. In other words, our goal is as follows:

To develop a sound texture synthesis algorithm, based both on existing synthesis algorithms and original ideas.

Presenting our research on the subject requires an overview of existing methods. But before moving to the presentation of the state of the art in this domain and detail our work, we need to clarify one last point: what should we aim for when building such an algorithm ? The answer to this question is useful in that it gives us the tools to compare existing methods on a common scale. To help us find an answer, let us review examples of potential applications for it and extract from those a set of explicit guidelines to our research.

1.2.1 Applications

Because they are usually found in the backgrounds of other sounds, a common usage of sound textures is to have them act as sonic wallpapers of another audio track. Used this way, they can create a sense of immersion in virtual environments.

Immersion

Sound textures (often referred to as "soundscapes" in this context, synonymous with the idea of a sonic landscape) convey information on what is happening around the listener without her/him needing to actually see any of those events, and saving the hassle of representing those events within the virtual environment.

In the field of video games, this can for instance be used to give life to completely fictional cities (convincing examples of which can be found in *The Witcher 3*, published in 2015 by CD Projekt RED [1]) or immerse the player in an immense natural environment (as is artfully done in *Journey*, published in 2012 by Thatgamecompany [2]). Sound textures are also omnipresent in movies, to the point where we would be hard pressed to find one (barring silent films) which discards them entirely: once again they are used as a compensation for what the watcher does not see and breathe realism into a scene, be it horrific or utterly common.

In both cases, a sound texture synthesis algorithm would need to be as convincing and as controllable in its output as possible so as to fit a given scene. If we once again take the example of rain, this could mean being able to control its density or the kind of surface it is falling upon, all the while keeping the result lifelike. In the case of video games, and since it is not possible to predict how much time the player will spend at a given point, the possibility of synthesizing any given length of sound would remove the need to loop a pre-recorded one (which often ends up being noticeable and irritating).

Artistic usage

Textures can also be used more freely as a mean of artistic expression. They can be found inside of multisensory installations, or transformed and diverted from their usual background role into a more central one (as is the case in *Synopsis As Texture* created by Florian Hecker in 2018).

It is difficult to predict exactly what each artist may want out of a synthesis algorithm, given both the subjectivity of the matter and their tendency to use such tools in inventive and unpredictable ways. Despite this we may say that having as much control over the algorithm as possible is desirable, so as to allow for inventive ways of manipulating it.

Compression

Alternatively, [Lu et al. 2004] suggests using synthesis algorithms as a form of compression: instead of storing the audio data of a textural sound, one could simply store the parameters used for its synthesis. Supposing that we are able to synthesize any length of texture, this would indeed be equivalent to compressing a texture into data of a fixed size with no regard for the initial length of the signal. Since this would simplify data transmission and overall facilitate its portability, this could be put to use in any of the aforementioned applications.

1.2.2 Research goals

From this list of potential applications we extract a set of guidelines for sound texture synthesis, and which in effect should be the objectives of any sound texture synthesis algorithm. For clarity's sake, we split them into five categories.

[1] See `https://thewitcher.com/en/witcher3/`
[2] See `http://thatgamecompany.com/journey/`

Realism

First and foremost comes realism: the synthesized texture needs to be as lifelike as possible for all uses that can be made of it. This is the main criterion on which synthesis algorithms are judged and as such needs to be heavily emphasized. The ultimate judges of this are the listeners, and as such a listening test is required to properly judge of the realism of the synthesis.

Variability

Because textures are by essence random, it is not enough that the synthesized textures be realistic: each one also need to be different from the others, else there would be no fundamental difference between using the algorithm and merely duplicating an existing texture. In other words, the algorithm needs to possess an element of unpredictability, and for it to be apparent in its output.

Flexibility

Given the wide array of existing sound textures and the diversity of their potential uses, a synthesis algorithm would strongly benefit from being able to synthesize the biggest possible range of textures without sacrificing any of their realism. Because textures are composed of small indiscernible and random events but also of occasional recognizable events, both harmonic (e.g. birds chirping) and not (e.g. crowd clapping), this goal can prove a serious challenge and is often harmful to the quality of the synthesis.

Extensibility

As mentioned in section 1.2.1, most applications would benefit from a mean of synthesizing an indefinitely long texture. This mean either being able to seamlessly link chunks of synthesized sound textures together, or fluidly creating the texture "on the fly". If coupled with a light enough algorithm, this extensibility means that texture could be created *ad lib* and in real-time.

Control

Last but not least, having control over the synthesis algorithm is crucial for all mentioned applications: depending on the context, this could mean being broadly able to tune the parameters of the synthesis algorithm, or being able to control higher-level parameters of the synthesized textures (such as the physical properties of the events they represent). In the best case, those parameters could be controlled continuously throughout the synthesized texture.

1.3 Overview of this thesis

This thesis is organized as follow:

- In Chapter 2, the links between sound textures and visual textures are explained, as well as the main notions required to understand them.
- In Chapter 3, the state of the art in sound texture synthesis is detailed.
- In Chapter 4, a perceptual texture synthesis method is introduced and our attempts at improving it are listed and explained.

- In Chapter 5, a visual texture synthesis method is introduced and our initial adaptation of it to sound texture synthesis is presented.
- In Chapter 6, several design choices in this algorithm are investigated to result in the final version of our synthesis method.
- In Chapter 7, the quality of this synthesis is evaluated and compared to that of other state of the art algorithms.

References for chapter 1

Athineos, M. and Ellis, D. P. (2003). "Sound texture modelling with linear prediction in both time and frequency domains". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cit. on pp. 2, 40).

Lu, L., Wenyin, L., and Zhang, H.-J. (2004). "Audio textures: Theory and applications". In: *IEEE transactions on speech and audio processing* (cit. on pp. 6, 38).

Saint-Arnaud, N. (1995). "Classification of sound textures". PhD thesis. Massachusetts Institute of Technology (cit. on p. 2).

Saint-Arnaud, N. and Popat, K. (1995). "Analysis and synthesis of sound textures". In: *in Readings in Computational Auditory Scene Analysis* (cit. on pp. 4, 38).

Chapter 2

Links between audition and vision

Chapter overview

Several recent sound texture algorithms are heavily inspired by existing methods in the visual domain, notably methods using convolutional neural networks. By displaying the evolution of their frequency content along time as a 2D matrix, sounds may be interpreted as images. However, knowledge transfer from the visual to the audio domain has to be done cautiously: while such representations of sound textures may be broadly assimilated to visual textures, a number of key differences still exist between the behaviors of the two.

Contents

Research fields are not impermeable to one another: many applications, initially aimed at a problem specific to one field are then generalized and used in others. This is especially true for the audio and visual research fields. In recent years the trend has mostly been for visual applications to be adapted and applied to audio, even more so with the democratization of deep learning methods. Tools like convolutional neural networks (CNN), first introduced for image recognition in [LeCun et al. 1998], are now used for dominant melody extraction ([Doras et al. 2019]) or voice anonymization ([Cohen-Hadria et al. 2019]).

This is especially true of sound texture synthesis: many of the methods presented in the state of the art of this thesis are inspired from equivalent methods for visual texture synthesis. Thoroughly presenting those methods thus requires introducing notions that are at the crossroad of the audio and visual domains. This is what the current chapter does: it introduces time-frequency representations, which allow sounds to be interpreted as monochrome images, and gives a quick presentation of visual textures and CNN while establishing the conventions and nomenclature used throughout this thesis.

2.1 Time-frequency representation

On a basic level, sound is represented as a waveform: this waveform can be interpreted as the position over time of a loudspeaker membrane producing said sound, and is a continuous signal with respect to time. When using digital signal processing (as is the case for all works mentioned in this thesis), this continuous signal is sampled at a given sampling frequency to obtain the digital waveform. Waveforms are 1-dimensional[3] temporal representations of sounds, and as such cannot be interpreted as images. The switch from this representation to a 2-dimensional one is performed using the Fourier transform.

2.1.1 Discrete Fourier Transform

The Fourier transform, named after Jean-Baptiste Joseph Fourier (1768-1830), is one of the cornerstones of signal processing. In essence, it is the decomposition of a time-domain signal into a sum of sinusoids of different frequencies. Because the human ear is sensible to those frequencies, this transform is omnipresent in audio.

Definition

As we are working on digital data, the signals we work with are discrete and finite. As such, the version of the Fourier transform we use is the discrete Fourier transform (DFT). This transform can be mathematically described as the projection of a complex vector x of length $N \in \mathbb{N}$ onto a basis of discrete sinusoids, and is performed as follows:

$$X_k = \sum_{n=0}^{N-1} x_n \times e^{-i\frac{2\pi kn}{K}} \quad \text{with} \quad k \in [0, K-1] \tag{2.1}$$

[3] At least in the case of monophonic sounds, to which this work is limited.

with X the discrete Fourier transform (the term is used interchangeably for the operation and its result) of x. n and k are respectively the time and frequency indices, and K is the length of X. In the same way that x_n is n^{th} sample of x, X_k is called the k^{th} bin of X.

In general, $K \geq N$: choosing $K > N$ is equivalent to zero-padding the signal x (by artificially adding 0 values to it until it reaches a size K), and as such does not increase the information contained in X. For this reason and unless stated otherwise, all Fourier transform mentioned in this work are performed using $K = N$ (meaning x and X are of same length).

It is also important to note that if x was sampled from a continuous time-signal, the frequency indices k correspond to frequencies f (in Hertz) in the continuous time-domain following:

$$f = f_s \frac{k}{K} \tag{2.2}$$

with f_s the sampling frequency.

X is a complex-valued signal: its squared absolute value $|X|^2$ indicates how the energy of the signal is spread along the frequency axis, while its phase $\angle X$ tells of the relative positions of the sinusoids.

In practice the DFT is computed using Fast Fourier Transform (FFT) algorithms. Because of this, the names FFT and DFT are often used interchangeably. Because of the way the FFT is usually implemented, the length N of the time signal is often chosen to be a power of 2 to fasten computations.

Properties

Out of the properties of the DFT and for clarity's sake, we only detail the few that are of use for the explanations contained in this thesis (see [Smith 2007b] for a thorough and complete overview of the DFT):

- The DFT is a linear transform, and as such the DFT of a sum of signals is the sum of their respective DFTs.
- If x is real-valued, as is often the case, X is conjugate symmetric meaning:

$$X_{K-k} = \overline{X_k} \tag{2.3}$$

  with the overline denoting the complex conjugate. As such, and given that K is in effect always a power of 2, we only need the first $\frac{K}{2} + 1$ elements of X to describe it entirely.

Inversion

The DFT is invertible, meaning it is possible to retrieve x from its DFT X. This inversion is given by:

$$x_n = \frac{1}{K} \sum_{k=0}^{K-1} X_k e^{i\frac{2\pi kn}{K}} \quad \text{with} \quad n \in [0, N-1] \tag{2.4}$$

2.1.2 Short-time Fourier transform and spectrograms

The DFT of a signal allows us to visualize how its content is spread along the frequency axis. But because the frequency content of most signals evolves with time, this visualization can quickly become too confused to be meaningfully interpreted when used on long signals. The short-time Fourier Transform (STFT) is a 2-dimensional representation of a temporal signal, along both time and frequency, developed to avoid this issue.

Definitions

The fundamental idea of the STFT is to split a time signal into a series of (potentially overlapping) chunks of a given size, to compute the DFT of those chunks and then stack them. This process outputs a 2D vector that gives us a view of the evolution of the frequency content of the signal. It is given by:

$$X_{k,m} = \sum_{n=0}^{L-1} w_n x_{mh+n} e^{-i\frac{2\pi kn}{K}} \quad \text{with} \quad k \in [0, K-1], m \in [0, M-1] \tag{2.5}$$

with L the length of the chunks and h the hop-size, i.e. the distance between the starting points of two neighboring chunks. This means that if $h < L$ those chunks are overlapping with each others by $L - h$ samples. As such, the overlap ratio is defined as $\frac{L-h}{L}$. w is an analysis window of size L that is applied to each chunk so as to avoid border effects in the DFT, which appear when the signal starts and ends on different values. m is the index of the time chunk, referring to a portion of the time signal comprised between mh and $mh + L$. The upper limit of m, M, is given by the floor value of $\frac{N-L}{h}+1$. The computation of the STFT is illustrated on Figure 2.1.

A column of X is called a frame and represents the DFT of a portion of x, while a line represents the evolution of a particular frequency bin over time. As with the DFT, the STFT outputs complex values (i.e. $X \in \mathbb{C}^K \times \mathbb{C}^M$).

But because X is complex, it is often replaced by its absolute value for representation purposes:

$$S_{m,k} = |X_{m,k}| \quad \text{with} \quad k \in [0, K-1], m \in [0, M-1] \tag{2.6}$$

This new matrix S is called spectrogram, and (as is apparent in the upcoming state of the art) is often used as a way to apply vision-based methods to sound by interpreting the values of each bin of the spectrogram as the intensity of a pixel in a monochrome image. While it gives an indication of the energy contained at given point in time and frequency, it is important to note that the spectrogram completely discards the phase of the STFT from which it was computed. It is common to represent spectrograms as 2D images with the amplitude being displayed on a color scale.

In this thesis, all time-frequency representation follow the convention of having time as horizontal axis (from left to right) and frequency as vertical axis (from bottom to top) when shown as images.

Properties

As for the DFT, we now detail the properties of the STFT and of spectrograms that are used throughout this thesis:

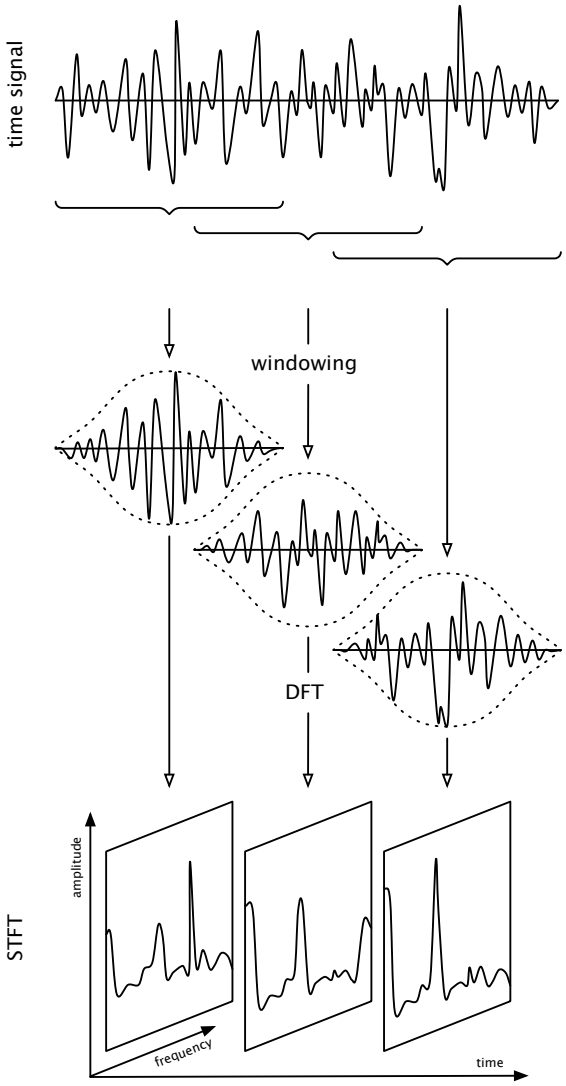


Figure 2.1: *Computation of the STFT: the time signal is first split into (potentially overlapping) windowed signals, which DFTs are then computed and stacked. The resulting STFT stores the evolution over time of the frequency content of the time signal.*

- The STFT is a linear transform, and as such the STFT of a sum of signals is the sum of their respective STFTs. Strictly speaking, this is not true for spectrograms: the triangle inequality states that the spectrogram of a sum of signals will be smaller at all points than the sum of their spectrograms. In most cases, however, this linearity may be approximated as long as the phases of the signals are not too strongly correlated.
- If x real-valued, as is often the case, X is conjugate symmetric along the frequency axis while S is symmetric, meaning:

$$X_{m,K-k} = \overline{X_{m,k}} \tag{2.7}$$

$$S_{m,K-k} = S_{m,k} \tag{2.8}$$

  Given that we take $K = N$ and that N is in most cases a power of 2, we only need the first $\frac{N}{2} + 1$ frequency bins of X and S to describe them entirely.

Inversion of an STFT

Just like the DFT, the STFT is invertible. Under the condition that the analysis window used w verifies the constant overlap-add (COLA) property at hop-size h, its inverse is straight-forward to compute. This property is met if:

$$\sum_{m=-\infty}^{\infty} w_{n-mh} = 1, \forall n \in \mathbb{Z} \tag{2.9}$$

This implies that adding and overlapping the windowed chunks taken from the signal x perfectly reconstitutes it. As such, it is possible to perfectly recover x from X by inverting each frame, and overlap-adding the resulting vectors. This process is illustrated in Figure 2.2.

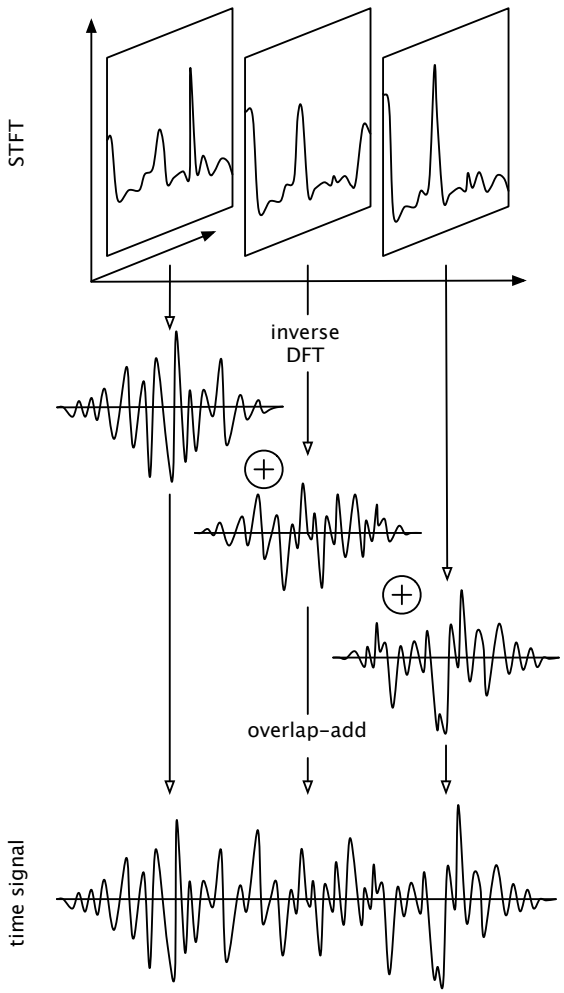


Figure 2.2: *Inversion of the STFT: each frame of the STFT is inverted using the inverse DFT, and is then overlap-added with the others to reconstruct the time signal.*

Consistency

While spectrograms can be interpreted as monochrome images, it is important to keep in mind that not all images can be considered as spectrograms. In a broader way, not all complex matrices can be considered as the STFT of an existing signal. Intuitively, this comes from the fact that as soon as the windows of the STFT overlap

(which in practice is generally the case), the content of the chunks are correlated. Because the end of a chunk is also reproduced at the beginning of the next one, their spectral contents are slightly similar: this means that the values of a frequency bin over the time cannot vary to abruptly.

This can be illustrated by a simple example, shown on Figure 2.3. One can imagine an artificial STFT of overlap 50% that is null everywhere except for a single frequency bin, which alternates between 0 and 1 every frame. During the inverse DFT phase of the inversion of the STFT, time chunks are recreated: in this case, empty chunks will alternate with ones containing mono-frequency sine waves. But because of the overlap 50% of the STFT, the recreated signal will not contain any null parts: for this reason, computing the STFT of this signal will not reproduce the original artificial STFT (this thought experiment voluntarily ignores the matter of phase to simplify its illustration, but this concept stays true even when the phase is taken into account).

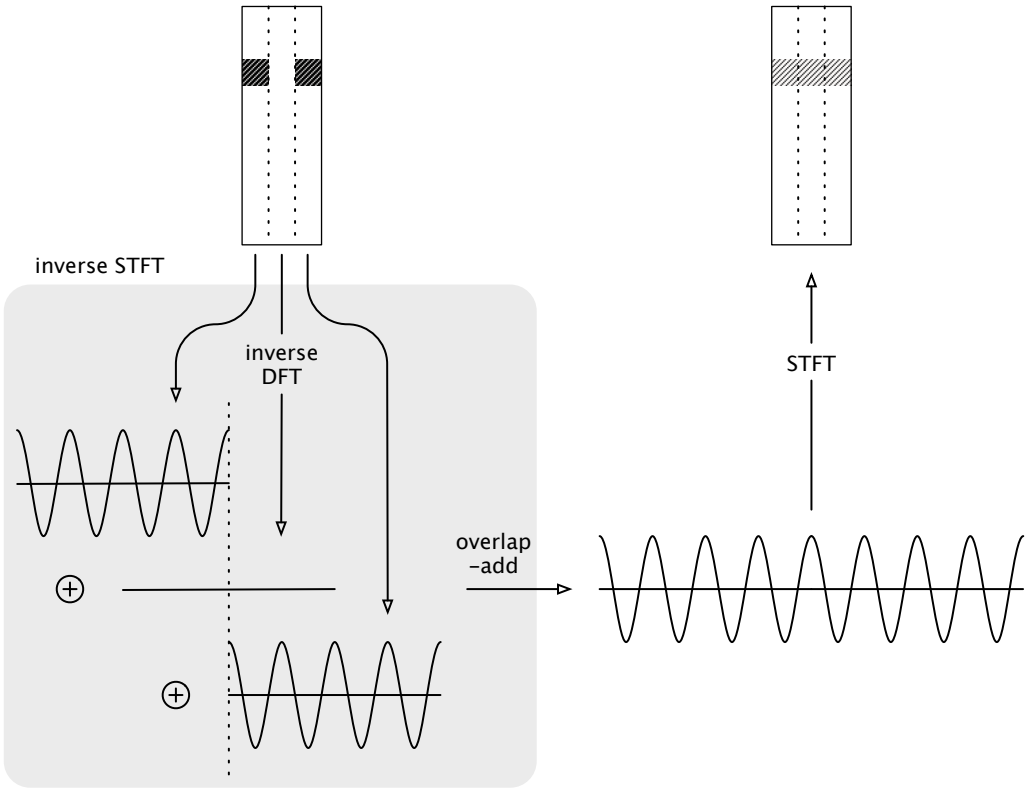


Figure 2.3: *Thought experiment where an artificial STFT of overlap* 50 *%, empty except for a row alternating between* 0 *and* 1*, is inverted to produce a time signal. The STFT of this signal is computed, but does not reproduce the same matrix as the initial one due to the overlap.*

Given some STFT parameters, a matrix having the property of "being the image of a signal by STFT" is called consistent. By extension, we call consistent a spectrogram which is the magnitude of a consistent complex matrix. This concept is crucial when synthesizing sounds via their STFT or their spectrogram, for instance by drawing one: simply producing an image does not guarantee that a sound possessing such a STFT or spectrogram exists.

Inversion of a spectrogram

Since spectrograms discard the information contained in the phase of the STFT, straightforwardly inverting each frame as we do for the inverse STFT proves impossible. Because of the issue of consistency discussed in the previous section, it is also not possible to simply decide on an arbitrary phase to associate to the spectrogram. Supposing that the spectrogram is consistent, it is thus necessary to retrieve a phase to recreate its consistent STFT. A classic method of doing so is the Griffin-Lim algorithm introduced in [Griffin et al. 1984]. If we are trying to invert a spectrogram matrix S, the algorithm can be described as follow:

1. Attributing a random phase matrix Θ_0 to S: $\tilde{X} = S\exp(i\Theta_0)$
2. Inverting the resulting STFT[4] $\tilde{X}$: $\tilde{x} = \text{STFT}^{-1}(\tilde{X})$
3. Computing the phase of $\text{STFT}(\tilde{x})$: $\tilde{\Theta} = \angle\text{STFT}(\tilde{x})$
4. Attributing the phase matrix $\tilde{\Theta}$ to S: $\tilde{X} = S\exp\left(i\tilde{\Theta}\right)$
5. Repeating steps 2. to 4. until $\tilde{X}$ is as close as desired to $\text{STFT}(\tilde{x})$

It is important to keep in mind that this method is an approximation, and the resulting inverted signal may present some artefacts. Those artefacts are particularly noticeable when trying to invert brief impacts which require very synchronous phases across frequencies, and may result in chirping sounds instead.

The Griffin-Lim algorithm is the most famous phase inversion algorithm, but is not the state of the art. Other algorithms such as those presented in [Le Roux et al. n.d.] and [Perraudin et al. 2013] have been introduced since, but mostly improve the speed of convergence without noticeably improving the quality of the reconstructed time signal.

2.1.3 Mel-spectrograms

Although the spectrogram is the most commonly encountered time-frequency representation, others exist. In particular, some representations aim at mimicking more closely the processing of sound performed by the human ear. For instance, and because of how the cochlea (a part of the inner ear) is designed, we are better capable of discriminating two close frequencies when they are low-pitched. Since the DFT has a uniform resolution across the frequency axis (in the sense that the distance between the frequencies of two neighboring bins is always the same), this means that some bins represented in the higher frequencies of the spectrogram are needlessly redundant and not perceptible while others at lower frequency are being overlooked.

[4] In most cases, $|\text{STFT}(\tilde{x})|$ and S are not equal.

To remedy this, it is possible to re-scale the spectrogram by evenly spacing the frequency bins along another frequency scale. Introduced in [Stevens et al. 1937], the Mel scale is a commonly found example of such a perceptual scale. Although there are several explicit formulae approximating it, the more popular one (introduced in [Makhoul et al. 1976]) is given by:

$$M(f) = 1125 \ln\left(1 + \frac{f}{700}\right) \tag{2.10}$$

with M the Mel frequency in Mels and f the frequency in Hertz.

The re-scaled spectrogram, called the Mel-spectrogram, presents the frequency data in a way that is more pertinent for tasks related to human hearing (such as sound synthesis). In practice, the Mel-spectrogram is obtained by filtering the spectrogram by a filter-bank containing filters evenly spaced along the Mel scale, such as the one shown in Figure 2.4: the filtering is done by performing a matrix product between the spectrogram of the analyzed signal and a matrix containing the filter bank.

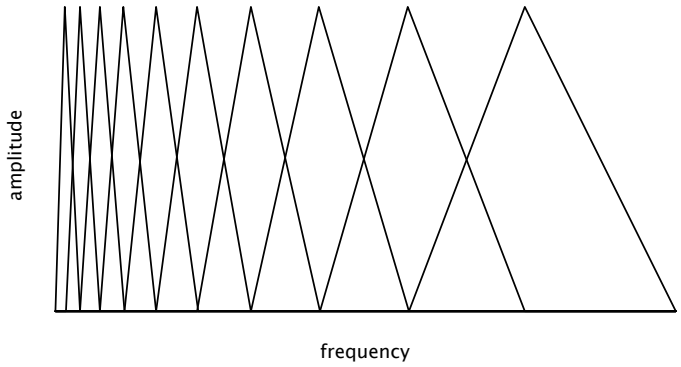


Figure 2.4: *Example of the amplitude of a 10-bands Mel filter-bank plotted along a linear frequency axis: the higher the center frequency, the larger the filter is. This results in a compression of the higher compared to the lower ones.*

To give an idea of the differences between spectrogram and Mel-spectrogram, two of them are put side-by-side in Figure 2.5. In this figure, it is visible that Mel-spectrograms accentuate the lower frequencies while squeezing the higher ones compared to the linear frequency representation of spectrograms.

In a similar fashion, other perceptual filter-banks choices can be made: for instance, it is possible to use the Bark scale (introduced in [Zwicker 1961]) instead or the equivalent rectangular bandwith (ERB) scale.

2.1.4 Constant Q transform and Wavelet Transform

Although none of those transforms are used in the work presented in this thesis, both the Constant Q Transform (CQT) and the Wavelet Transform deserve to be mentioned so as to help situate our work among existing ones. Those transforms can be interpreted quite similarly to the STFT as time-frequency representations, but aim at getting rid of the constant resolution of the STFT along the frequency axis. In the case of the CQT, the sinusoids on which the time signal is projected are

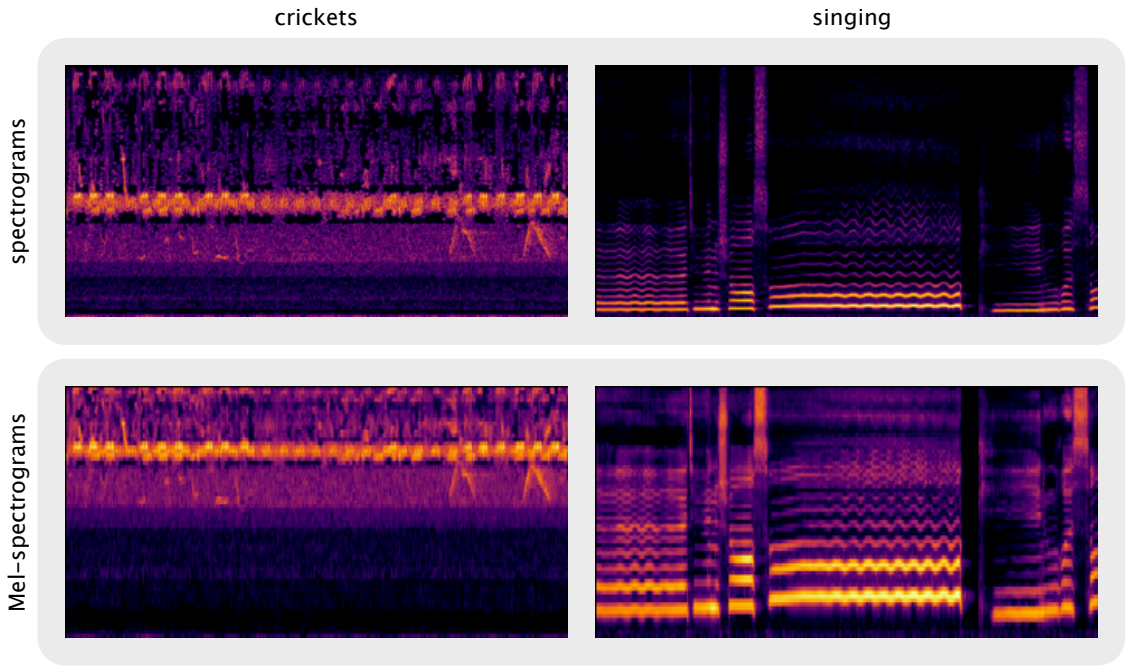


Figure 2.5: *Side-by-side comparison of the spectrograms (up) and Mel-spectrograms (down) of two sounds: a recording of crickets chirping (left) and a singing voice recording (right). Both representations are shown using a logarithmic scale on their amplitudes.*

replaced by windowed sinusoids which lengths are chosen so as to keep a constant Q factor (defined as the ratio of their frequency over the width of the frequency peak of their DFT): this results in shorter sinusoids at higher frequencies. The Wavelet Transform can be seen as extension of the CQT, replacing the windowed sinusoids by more general kernels called wavelets, and parametrized by a scale factor.

2.2 From visual to sound textures

The array of time-frequency representations showcased in the previous section is often used as a bridge between the visual and the audio fields. By representing an audio signal as a monochrome image, it is indeed possible to apply visual signal processing methods to sounds. This knowledge transfer is even common enough to be criticized, such as in [Pons et al. 2016], where the author advocates for more reflection over the difference between a time-frequency representation and an image when trying to tackle an audio-related task.

In our case, the counterpart to sound texture synthesis rather obviously is visual texture synthesis. But directly using a method created to synthesize visual textures in order to synthesize time-frequency representations of a sound textures implies that those representations can be considered as visual textures themselves. The following sections aim at brushing a clearer picture of what visual textures are, and at exploring their similarities with time-frequency representations of sound textures.

2.2.1 Definition

As is the case for sound textures, there exists no precise definition of visual textures. Instead, and as detailed in [Zhou 2006], several have been proposed. In light of our adopted definition of sound textures, one of those visual texture definitions looks familiar: "an organised area phenomenon which can be decomposed into primitives having specific spatial distributions" ([Haralick et al. 1979][5]). On the visual texture examples given in Figure 2.6, those primitives can be patches of fur, of ground or of vegetation.

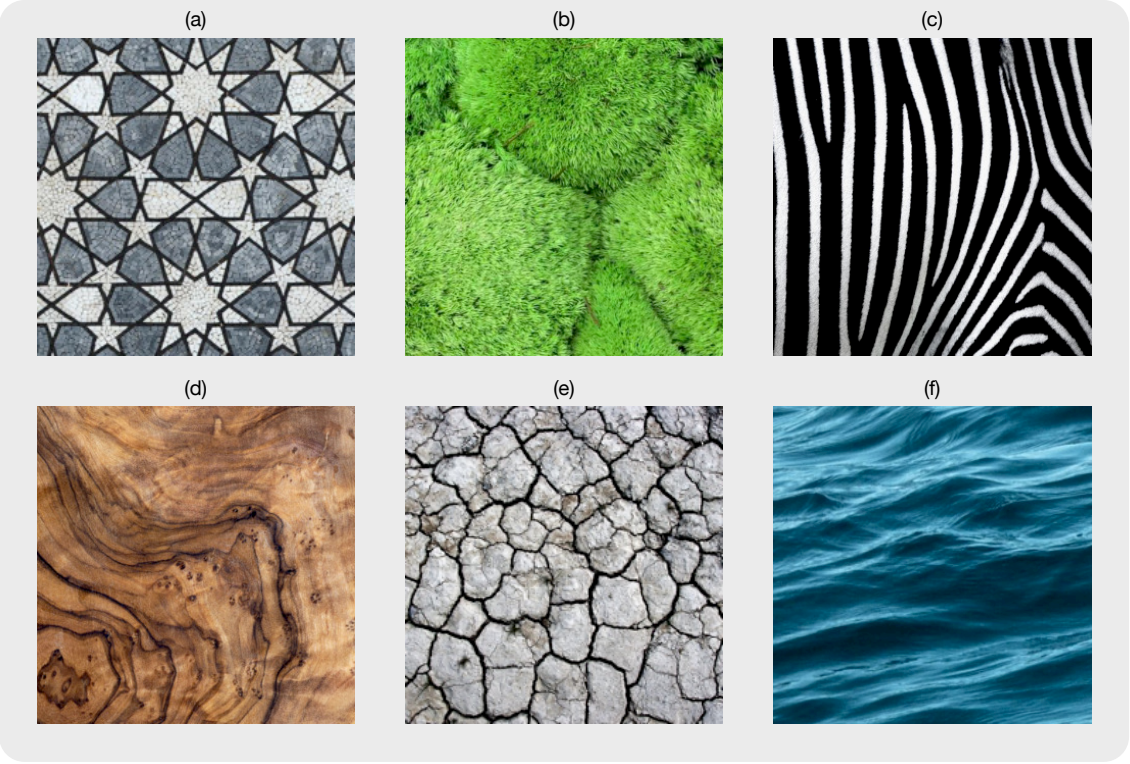


Figure 2.6: *Examples of visual textures: (a) arabesque pattern, (b) wild moss, (c) zebra fur, (d) wood grain, (e) dried earth and (f) water surface.*

Although Zhou furthers this definition by crossing it with others coming from diverse sub-fields of vision and by separating textures into categories, Haralick's definition suffices for the needs of our comparison.

2.2.2 Comparison

Having introduced visual textures, it is possible to try and answer the following question: is the time-frequency representation of a sound texture a visual texture itself ? To help with this comparison, the spectrograms of a few sound textures are shown on Figure 2.7: despite spectrograms being showcased, the following discussions stays true for other time-frequency representations.

[5] One may notice the fact that founding works on visual textures predate those on sound textures by more than a decade, which further justifies why one might try to carry some of the more recent research on sound texture synthesis over to the audio field.

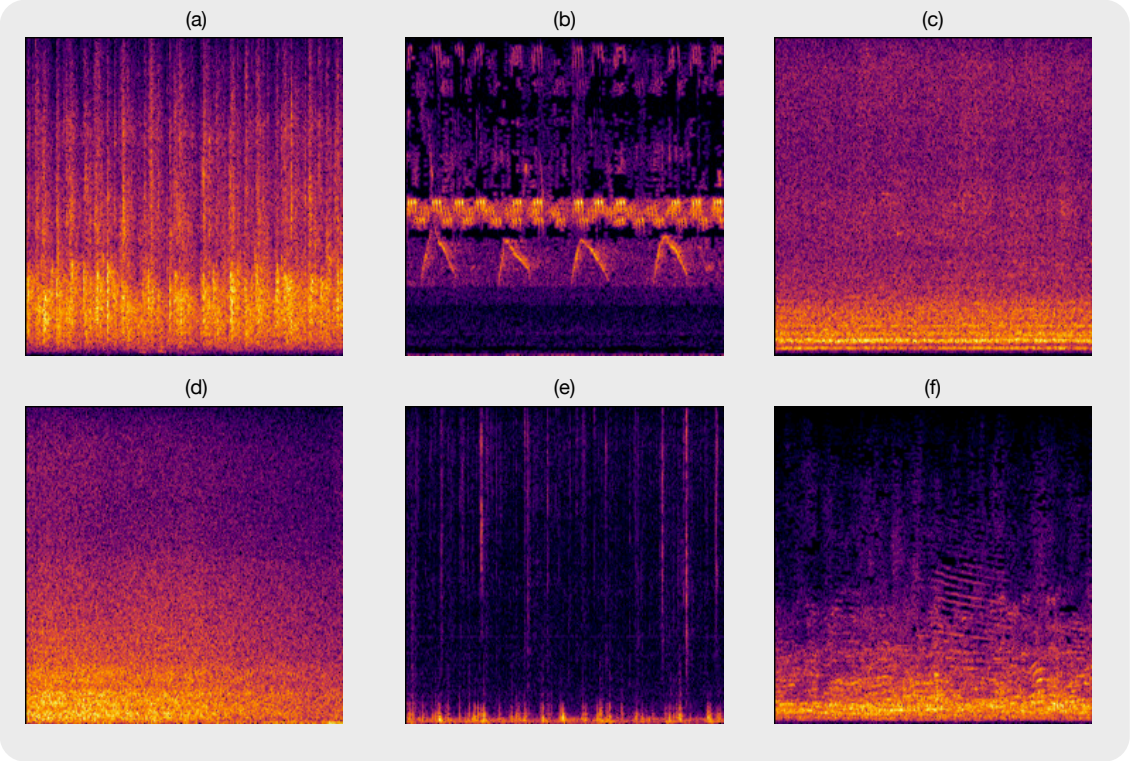


Figure 2.7: *Examples of sound texture spectrograms: (a) applause, (b) crickets, (c) bees, (d) waves, (e) fire and (f) crowd.*

Similarities in definition

A common point between both our adopted definition of sound texture and the presented definition of visual texture is the presence of an elementary component. Be it called atom or primitive, this element is the basis of the texture and it is its organization following a higher-order distribution that globally forms the texture. But although this similarity may seem encouraging, it is in the time-frequency domain that we need to compare sound atoms to visual primitives.

Since most sound atoms are either short percussive events, harmonic events or noisy events their representation in the time-frequency domain will respectively either be vertical segments (e.g. the claps in spectrogram (a) of Figure 2.7), horizontal curves (e.g. the inverted "V" pattern of crickets in spectrogram (b) of Figure 2.7) or noisy patches (e.g. bees humming in spectrogram (c) of Figure 2.7). Under this assumption, and since those patterns periodically reappear along the time axis, those sound atoms can indeed be interpreted as visual primitives when represented in the time-frequency domain. The temporal distribution of their apparitions translates to a time-frequency distribution, although only affecting the horizontal axis: as such, a sound texture may be considered as a visual texture in the time-frequency domain.

Differences in behavior

Despite this similarity, it is important to keep in mind that those two kind of textures are intrinsically different. A key notion where both differ is in their invariances (by which we mean transformations which, when applied to a texture, output textures of

a similar kind).

In sound textures, atoms are organized temporally: since this organization is stable throughout time, this means that sound textures are invariant to time translations. In the time-frequency domain, this translates to an invariance to horizontal translations. Conversely, vertical translations correspond to pitch shifts, which effect on textures we cannot predict. For visual textures, and since both axes represent a spatial dimension, the invariance exists indiscriminately in both directions. In addition to this, while visual textures may be rotated without harming their content, a rotation does not make any sense for the time-frequency representation of a sound texture: such a transformation implies that the axes of the representation are homogeneous, which is not the case here.

Because of the properties of the frequency domain, sound textures possess characteristics that have no equivalent in visual textures. In particular, frequency bins can be strongly correlated to other non-adjacent bins via the underlying physical properties of its source. An example of such a phenomenon is the presence of harmonics. Those harmonics, visible for instance on the right row of Figure 2.5 or on spectrogram (f) of Figure 2.7, can be defined as frequencies appearing at multiples of a fundamental frequency: visually, this translates into the presence of several vertical copies of the fundamental frequency line. This means that while in visual textures close pixels are strongly correlated, correlations in the time-frequency representation of sound textures are potentially less local.

The two kind of textures also react differently to the addition of atoms/primitives. Because the computation of a spectrogram can be approximately considered linear, the different contributions of each atoms are summed on the final spectrogram of the texture. In the case of visual textures though, each primitive may hide parts of others present primitives, which results in competing contributions to the texture.

This implies that although sound textures may be assimilated to visual textures when represented in the time-frequency, their properties differ enough that directly transferring a visual synthesis algorithm to audio would be poorly effective. This intuition is even further substantiated throughout the upcoming state of the art in sound texture synthesis.

2.3 Convolutionnal neural networks

The presentation of said state of the art, in addition to the work presented in this thesis, requires us to briefly introduce another tool initially developed for the visual domain and now widely used in audio: convolutional neural networks (CNNs), a specific kind of neural network.

2.3.1 Introduction to neural networks

Neural networks have been used in a wide array of tasks, ranging from image classification ([He et al. 2016]) to singing voice synthesis([Blaauw et al. 2016]), but also photo-realistic face generation ([Karras et al. 2019]) or music classification ([Choi et al. 2017]). On a basic level, neural networks are algorithms that learn and evolve

in order to improve their ability to perform a designated task. Taking the example of musical style classification, this is done by first gathering a dataset of annotated inputs: here this means selecting a set of music tracks and choosing a representation for them (waveform, spectrogram, etc.), and then associating a tag ("rock", "pop", "rap", etc.) to each of them. We then choose an architecture, which dictates how the input is processed to create the output, and initialize its parameters. Since the initialization of the parameters is often random, this means that the outputs of the network, in our example the tags, are at first mostly wrong. For this reason, the network undergoes a training phase: using the annotated dataset, the parameters of the network are optimized so as to reduce its mistakes. If this phase is successful the network is then able to not only perform its task successfully on the training dataset, but also to generalize it to inputs it never trained on.

Although it is tempting to label neural networks as "magical black boxes", their usage (as those things often go) is in practice complex. The datasets their training requires are often hard to come by for tasks that are not common. This, coupled with the fact that the training of a network is often computationally expensive, is one of the main reasons neural networks have only found widespread success in the past decade. But more so than the dataset, the architecture of the network is key: in order to be able to generalize its process, the architecture needs to be adapted to the task at hand. The designing of a network is often a complex work, and may require many trials and error. Last but not least is the optimization of the weights in itself: although mathematical optimization is a well-studied field, the presence of potentially millions of parameters to be tuned makes minimizing the errors of the network an arduous process.

While we do not have the pretension of giving an exhaustive explanation of neural networks, the following sections aim at introducing the concepts required for the understanding of convolutional neural networks, which are a recurring theme in this thesis.

2.3.2 Layers

So as to give a concise presentation of CNNs, we adopt a simple view of neural network architectures as a sequential succession of layers: while the first layer processes the input of the network, all following layers process the output of the previous one. Those networks are called feed-forward networks, in that data is always fed to the next layer without being sent backward. Let us now detail the common layers encountered in neural networks, and CNNs in particular.

Dense layer

Dense layers are composed of individual units called neurons, which are based on a computational (and simplified) interpretation of how biological neurons work. In this interpretation, neurons fire off a signal once the sum of their input reaches a given threshold. An artificial neuron, as illustrated in Figure 2.8 processes incoming numerical data by multiplying each of its K input x_k by its associated weight w_k and summing the results. The sum is then shifted by a bias b and a non-linear function f is applied to it to obtain the output y. Several choices of non-linear function exist, and among them the most popular are:

- the sigmoid function: $\sigma(x_k) = \frac{1}{(1+\exp(x_k))}$
- the hyperbolic tangent function: $\tanh(x_k) = 2\sigma(2x_k) - 1$
- the Rectified Linear Unit (ReLU): $f(x_k) = \max(0, x_k)$
- the softmax function: $f(x_k) = \frac{\exp(x_k)}{\sum_{n=0}^{K} \exp(x_n)}$

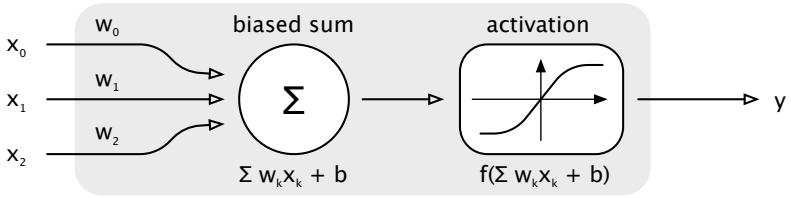


Figure 2.8: *Illustration of the processing performed by a neuron (in grey): incoming numerical values (x_0, x_1 and x_2) are multiplied by their respective weights (w_0, w_1 and w_2), the sum of those products is shifted by b and a non-linear function f is applied to it. The result y is the output of the neuron.*

A dense layer is the parallel association of neurons processing a common numerical input, but each having its independant weights and bias. Such a layer is shown in Figure 2.9.

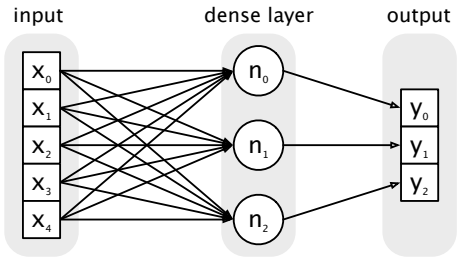


Figure 2.9: *Dense layer composed of 3 neurons processing a common input x_k.*

The idea behind dense layers is for each neuron to specialize itself during the training phase, with its weights and bias being modified until the output of the neuron is maximized for a specific pattern of values in the input. While being fundamental in neural networks, those layers present several disadvantages. Each neuron being linked to all of the inputs via independent weights means that the layer completely discard any information regarding the relative position of each individual input: no importance is given to the fact that two inputs may be spatially close in an input image, for instance. Because of this the inputs to dense layers are commonly represented (or flattened if need be) as 1D vectors. Since in many cases those inputs (which can for instance be the values of the pixels of an image or of the samples of a waveform) are strongly correlated to their neighbors, an important quantity of information is discarded by dense layers. Even worse is the fact that any spatial operation on the input (such as translations or rotations) corresponds to a nonsensical shuffling of the input from the point of view of the layer, and as such the layer is not robust to them. This, coupled with the fact that dense layers quickly reach sizes of millions of parameters when working with life-sized audio or visual data makes them poor candidates for working on images or sounds.

Convolutional layer

Convolutional layers solve the issues the dense layers have with big, spatially/temporally organized data. As the name hints at, those layers are the cornerstone of CNNs and are based on convolution (although those convolutions can be argued as technically being correlations). Unlike dense layers the inputs of convolutional layers are organized as 3D matrices, or by extension as 1D or 2D matrices. In the example of an image, we use the convention of having the first two dimensions represent the two spatial dimensions of the image, while the last one represents the different color channels.

Convolutional layers are composed of filters. Those filters are 3D matrices of parameters having the same depth as the input. In this thesis, the depth designates the last of the 3 dimensions. Each filters is convolved to the input by travelling along its first two dimensions. The result of this convolution can be expressed as a 2D matrix following:

$$F(x,y) = \sum_{a=0}^{k} \sum_{b=0}^{l} \sum_{c=0}^{m} X(xh_x + a, yh_y + b, c) f(a,b,c) \tag{2.11}$$

with F the output of the convolution, f the filter values, and X the input matrix (which can be zero-padded along its first two dimensions depending on the configuration adopted). The couple (h_x, h_y) indicates the size of the hops made between each convolution in both directions, and is called stride. (k, l) is the size of the filter along the first two axes. m is the size of the last dimension, common to X and f. This convolution is then followed by a non-linear activation function such as one of those listed in the previous section, with the most commonly used being the ReLU. The process is illustrated on Figure 2.10.

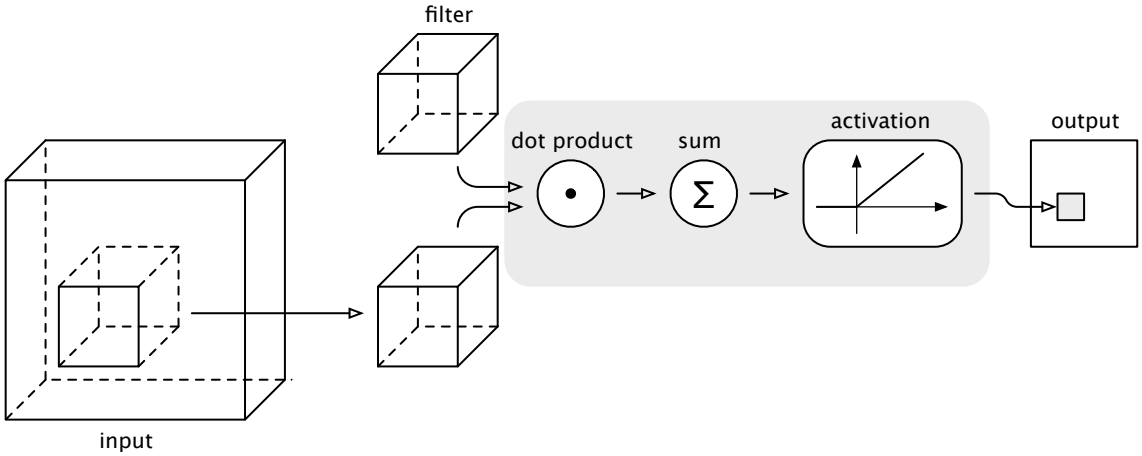


Figure 2.10: *Convolution of the input 3D matrix by a filter along its first two dimensions: a dot product between the filter and a part of the input is performed, followed by a global sum of the resulting matrix. The activation of the resulting value is computed and placed in the output matrix, at a position relative to that of the input excerpt. The process is repeated along the first two dimensions of the input.*

The 2D matrices resulting of their convolutions with the input matrix are called feature maps. Because the filters of a same layer share the same dimensions and

stride, those feature maps have identical shapes. They can thus be stacked along an added third dimension, which inherits the name "depth", so as to form a 3D matrix. Such a layer is shown in Figure 2.11.

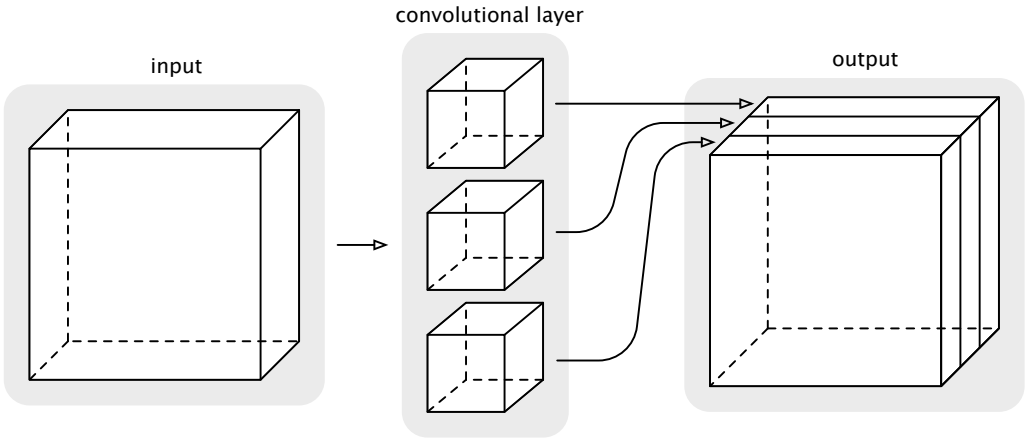


Figure 2.11: *Convolution of the input matrix by a series of filters: the resulting 3D matrix is called the feature map of the layer, and the index along its depth refers to the index of the filter that generated this slice of the output.*

The convolution between a filter and a part of the input matrix yields a high value at positions where both possess similar patterns. Because of this, the action of convolutional layers can be interpreted as each filter scanning the input for local patterns similar to theirs, and storing a similarity score inside its feature map. Contrarily to dense layers, in which each neuron is sensible to one global pattern over the whole input, convolutional layers are more flexible and adapted to spatially organized data: this is due to them scanning the input small area by small area. Because they systematically travel across the first two dimensions of the input, they are still able to detect a pattern if the content of the input is translated along those dimensions (although this is not the case for rotations). If they are correctly trained, each filter in a layer specializes itself so as to react to a particular local pattern, and potentially does so at any position in the input. Because the same weights of the filter are used across the input, convolutional layers are also lighter in parameter number compared to dense layers.

Down-sampling layers

So as to reduce the size of the data that passes through the convolutional network and feed the following layers a more global view of a feature map without requiring bigger and bigger filters, it is common practice to down-sample the feature maps. This process is usually performed by extracting a part of the input of a given size (k, l) along the first two axes of the input, applying a down-sampling operation to it, and repeating the operation with hops of size (h_x, h_y) along the first two dimensions. The two most common down-sampling operations are:

- max-pooling, where the value retained for each part of the input is its local maximum along the first two dimensions. This can be expressed as:

$$Y(x, y, z) = \max_{(a,b)\in[0,k]\times[0,l]} X(xh_x + a, yh_y + b, z) \tag{2.12}$$

with X the input of the layer and Y its output.

- average pooling, where the value retained for each part is its mean on the first two dimensions.This can be expressed as:

$$Y(x, y, z) = \frac{1}{kl} \sum_{a=0}^{k} \sum_{b=0}^{l} X(xh_x + a, yh_y + b, z) \tag{2.13}$$

with X the input of the layer and Y its output.

It is also possible to replace pooling layers by (trainable) convolutional ones with strides of (k, l). This is illustrated in [Springenberg et al. 2014] where results comparable to the state of the art in object recognition are obtained without using pooling layers.

In addition to reducing the size of the matrices being manipulated, pooling also helps making the representation of the data more robust. Since only the average or maximum of each small area is retained, small transformations won't affect the representation. Because of this, pooling layers are often associated with convolutional layers.

2.3.3 Standard classification architecture

Using those elementary layers it is possible to describe the working of a typical CNN used for a classification task, such the LeNet 5 network introduced for the first time in [LeCun et al. 1998]. This CNN is represented in Figure 2.12, and can be broken down into two parts.

Feature extraction

The first of those part aims at extracting a series of features from the input, and consists in a succession of convolutional and pooling layers. The first convolutional layer converts the input into a stack of feature maps representing the activations of its filters. Those feature maps are then down-sampled in order to give a broader view of their content to the following convolutional layer. While the first feature maps indicate the presence of a learned pattern in the input, the second indicate the presence of a learned pattern in the first (i.e. "patterns of patterns"). By chaining blocks of convolution and down-sampling, it is thus possible to describe both broader and more complex patterns (or features) in the input. The feature extraction part of the CNN is concluded by a flattening operation, simply consisting in the reshaping of the last feature maps into a 1D vector.

Classification

The second part of the CNN takes this representation of the high-level features of the input and passes it through several dense layers. This dense part of the network, also referred to as Fully Connected, is designed to learn a non-linear combination of the features while the output of its last layer is considered the output of the network. When used for a classification task, it is common to consider the output of each neuron of the last layer to represent the probability of the input belonging to one of the classes. So as to interpret those outputs as probabilities, and in the case of multi-class

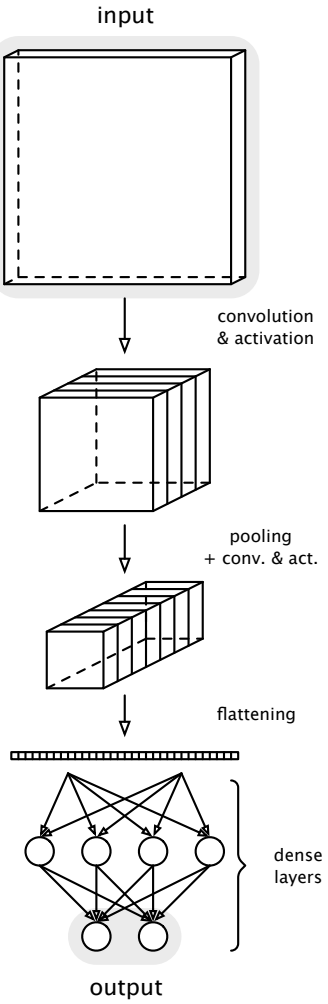


Figure 2.12: *Architecture of a typical CNN used for classification: a series of convolution and pooling layers first extracts features from the input matrix. After flattening it, the last feature map is used by the dense layers to predict a class to which the input belongs.*

classification (i.e. several classes are available, but only one may be chosen at a time), the softmax function can be chosen as the activation function of the last layer: the output of each neuron is then comprised between 0 and 1, and the sum of all outputs is 1.

This architecture is merely an example of a simple classification CNN. From there, one could modify the number of layers, the parameters of each convolutional (number and size of filters, stride, activation function), pooling (size and stride) and dense (number of neurons and activation function) layer. This is only a brief presentation of a standard architecture: although they are of no direct importance to this thesis, many different and more complex ones are available publicly (some using particularly inventive and clever tricks).

2.3.4 Training and optimization

While the architecture of the network is responsible for its ability to learn and generalize its knowledge, it is still necessary to train said network for there to be

any learning. The goal of the training phase is to shift the values of the parameters of the network in order to reduce its errors when performing the task it was given. Using the example of classification, this would mean being able to correctly predict the class of its input.

Parameters initialization

All parameters first need to be assigned an initial value. Setting all values to a given constant is usually a poor idea, as it makes the network perfectly symmetrical. This symmetry is a burden to the updating of the parameters, as those updates would also be symmetrical and thus no part of the network would specialize itself. This is why parameters are usually randomly initialized, either by using a random distribution (e.g. uniform or normal) or by using more advanced initialization techniques aimed at easing the learning phase (e.g. the Glorot initialization introduced in [Glorot et al. 2010]).

Loss function

All parameters of the network being initialized, the error the network makes when predicting the output associated to an input then needs to be quantified. This error is expressed using a loss function. If we denote y_k the k-th of the K element of the output y, and $\tilde{y_k}$ its expected value, some common loss functions are:

- the Mean-Squared Error (MSE) :

$$\mathcal{L}(y) = \frac{1}{K}\sum_{k=0}^{K}(y_k - \tilde{y_k})^2 \tag{2.14}$$

  This loss simply expresses the distance between predicted and expected values.

- the Cross Entropy :

$$\mathcal{L}(y) = -\frac{1}{K}\sum_{k=0}^{K}[y_k \log(\tilde{y_k} + (1 - y_k)) \log(1 - \tilde{y_k})] \tag{2.15}$$

  This losses expresses the divergence between the obtained and expected probability distributions when the expected value is a one-hot vector (i.e. all values of the vector are 0, except one that is set to 1).

Optimization

Once the error is quantified by a loss function, it becomes possible to use optimization methods to train the network. For each input of the labelled dataset, the loss function is computed and the parameters of the network are modified so as to minimize the loss[6]. This process may be repeated several times over the dataset, with each iteration being called an epoch of the training. Some examples of optimization methods are:

[6] In practice, and in order to avoid erratic (and possibly counter-productive) movements in the parameter space, those updates to the parameters are often performed so as to minimize the loss function of batches of inputs at a time.

- the Gradient Descent algorithm, which is the simplest form of optimization. In it, the gradient of the loss function $\mathcal{L}$ is computed with regard to the parameters of the network. Each value of the gradient is then subtracted to its corresponding parameter, after being multiplied by a factor α called the learning rate. Intuitively, this consists in updating the parameters of the network so as to go down the slope of the gradient of $\mathcal{L}$.
- the Adaptive Moment Estimation (Adam) algorithm, which adds several features to the gradient descent. Adam stores the exponentially decaying average of previous gradients and squared gradients. It uses the first instead of the gradient during updates, which gives a momentum to the values of the gradient, and the second to modify the individual learning rate of each parameter.
- the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, which is not as popular as others but still present in this thesis. This method is a quasi-Newton optimization algorithm, and stores values of the gradient of L in order to compute an estimate of its second-order derivative (Hessian matrix). The approximate Hessian is then used to find a root of the gradient, which leads to finding minima of the loss function. It is possible to only store the last values of the gradient instead of its whole history, which leads to the limited BGFS (L-BFGS) method, and to add box constraints to the solution, which leads to the boxed L-BFGS (L-BFGS-B).

Although some optimization methods (like Adam) were designed with neural network training in mind, it must be noted that they can still be applied in a general goal of modifying a given vector so as to minimize a target function, and are used as such in this thesis.

References for chapter 2

Blaauw, M. and Bonada, J. (2016). "Modeling and transforming speech using variational autoencoders." In: *Interspeech* (cit. on p. 23).

Choi, K. et al. (2017). "Convolutional recurrent neural networks for music classification". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cit. on p. 23).

Cohen-Hadria, A. et al. (2019). "Voice Anonymization in Urban Sound Recordings". In: *29th IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. Pittsburgh, Pennsylvania, USA (cit. on p. 12).

Doras, G., Esling, P., and Peeters, G. (2019). "On the use of u-net for dominant melody estimation in polyphonic music". In: *International Workshop on Multilayer Music Representation and Processing (MMRP)* (cit. on p. 12).

Glorot, X. and Bengio, Y. (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *13th International Conference on Artificial Intelligence and Statistics* (cit. on p. 30).

Griffin, D. and Lim, J. (1984). "Signal estimation from modified short-time Fourier transform". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* (cit. on p. 18).

Haralick, R. M. et al. (1979). "Statistical and structural approaches to texture". In: *Proceedings of the IEEE* (cit. on p. 21).

He, K. et al. (2016). "Deep residual learning for image recognition". In: *IEEE conference on computer vision and pattern recognition* (cit. on p. 23).

Karras, T., Laine, S., and Aila, T. (2019). "A style-based generator architecture for generative adversarial networks". In: *IEEE Conference on Computer Vision and Pattern Recognition* (cit. on p. 23).

Le Roux, J. et al. (n.d.). "Fast signal reconstruction from magnitude STFT spectrogram based on spectrogram consistency". In: (cit. on p. 18).

LeCun, Y. et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* (cit. on pp. 12, 28).

Makhoul, J. and Cosell, L. (1976). "LPCW: An LPC vocoder with linear predictive spectral warping". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cit. on p. 19).

Perraudin, N., Balazs, P., and Søndergaard, P. L. (2013). "A fast Griffin-Lim algorithm". In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (cit. on p. 18).

Pons, J., Lidy, T., and Serra, X. (2016). "Experimenting with musically motivated convolutional neural networks". In: *14th International Workshop on Content-Based Multimedia Indexing (CBMI)* (cit. on p. 20).

Smith, J. O. (2007b). *Mathematics of the discrete Fourier transform (DFT): with audio applications*. Julius Smith (cit. on p. 13).

Springenberg, J. T. et al. (2014). "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (cit. on p. 28).

Stevens, S. S., Volkmann, J., and Newman, E. B. (1937). "A scale for the measurement of the psychological magnitude pitch". In: *The Journal of the Acoustical Society of America* (cit. on p. 19).

Zhou, D. (2006). "Texture analysis and synthesis using a generic Markov-Gibbs image model". PhD thesis. ResearchSpace@ Auckland (cit. on p. 21).

Zwicker, E. (1961). “Subdivision of the audible frequency range into critical bands (Frequenzgruppen)”. In: *The Journal of the Acoustical Society of America* (cit. on p. 19).

Chapter 3

State of the art in sound texture synthesis

Chapter overview

A variety of paradigms currently exist for sound texture synthesis. Physics-based methods reproduce the physical process at the source of the texture to synthesize new sound textures of the same kind. Granular methods use audio grains taken from an original texture and rearrange them to produce a texture resembling the original. Parametric methods extract a set of parameters from the original and create new textures by matching those parameters. This thesis is centered around parametric synthesis method.

Contents

In the introduction to this thesis, five points central to sound texture synthesis were outlined: the realism of the outputs, the variability between them, the flexibility to produce different kind of textures, the possibility of extending a synthesized texture and the ability to control the synthesis algorithm. Those research goals can be seen as the general goals of texture synthesis methods, of which an overview is given throughout this chapter, and are a way of comparing them on even grounds.

While the term "sound texture" has already been investigated, it is the term "synthesis" that may need further clarifications. Although some of the methods presented in this chapter are proper synthesis methods, the vast majority are actually re-synthesis methods. By this, we mean that they work by analysis-synthesis: an existing sound texture signal is first decomposed and analyzed before being used to produce a new texture similar to it. Despite that, and because this distinction is often ignored in the field of sound texture synthesis, the term "synthesis" is also meant as "re-synthesis" in this thesis.

Out of existing syntheses, a prominent category is parametric[7] synthesis. Parametric methods aim at extracting a set of given parameters from the analyzed signal, and at recreating a synthesized signal possessing the same parameters. If the parametrization is well done, the synthesized signal belongs to the same texture category as the analyzed one while not being a simple copy of it. Because parametric methods are important to this thesis, we start by giving an overview of other methods that do not fit in this category.

3.1 Non-parametric texture synthesis

Although non-parametric methods are extremely diverse in their approaches to the synthesis of sound textures, it is possible to split them into groups that share a common paradigm.

3.1.1 Physics-based synthesis

Physics-based synthesis are rooted in the simulation of the physical phenomena at the origin of the texture. By properly approximating their behavior, it becomes possible to generate a texture by simulating the physical scene and extracting the sound signal generated by it. It can also be noted that those methods represent the main kind of "proper" synthesis: the signal generated is not the recreation of an existing texture but an original one.

In [Miklavcic et al. 2004], the impact of drops of rain on hard and water surface is expressed in order to reproduce the acoustic scene of a listener surrounded by drops of rain falling on both hard ground and puddles. A numerical simulation is then run by randomly generating a number of drops of varying sizes, directions and positions. Because all mathematical expressions used in this algorithm are analytical, this results in a computationally light synthesis method for the generation of rain sound texture. Since we could not find textures synthesized using this method, its

[7] Here "parametric" is not meant in the sense of "possessing parameters" (which is the case of all methods presented here) but is rather used to designate a particular sort of synthesis paradigm.

quality cannot be asserted.

In [Peltola et al. 2007], a simple model of excited resonant filter is used to simulate a single hand-clap. In order to simulate the noise generated by a crowd clapping, filters of varying resonances (which is the principal frequency at which a filter oscillates when excited) are assigned to each virtual clapper while exponentially decaying noise sources are used to excite them. Several global settings for clapper synchronization can be chosen. If no synchronization is desired, each one of them follows his own clapping frequency or claps at random, following a Poisson distribution. Conversely, the impulses of each clapper can be set to erratically try to follow a "model clapper". Despite this concept of synchronization being interesting, the presence of very high-pitched harmonics in the synthesized audio harm the realism of the synthesis.

In [Moss et al. 2010], a fluid simulator is coupled to a model of bubble in order to emulate the sound of a liquid being continuously poured into another one. Under the assumption that the main sources of sound are the bubbles, and that the interactions between them can be neglected, each bubble is considered as a harmonic oscillator excited at its creation. The fact that they can be non-spherical is taken into account using spherical harmonics. The generation and the radius distribution of the bubbles are taken from existing fluid simulations. The resulting algorithm is also able to simulate the sound of a flowing brook. Synthesized sounds are realistic and can easily be coordinated with a visualization of the fluid.

In [Chadwick et al. 2011], the combustion of fuel gas is simulated while modeling the density of gas and the temperature field. The fluctuation of density created by the sudden release of heat of the combustion then generates sound waves that resemble the low-pitched rumbling of large flames. In order to bring realism to this sound, a method which adds synchronized higher-pitched noise to the rumbling is presented. The combustion can also be synchronized with visual fire animations for a convincing audiovisual result.

In [Oksanen et al. 2013], a jackhammer is modelized as a solid steel bar excited by series of impulses. The longitudinal and transversal vibrations resulting from this excitation are summed in order to obtain a sound output. To make up for the absence of sound propagation through the air in this model, a high-pass filter is used on the sound output. The resulting audio signal emulates the impact a jackhammer: although the showcased example is lackluster, this may be attributed to a too strong tuning of the high-pass filter.

Physics-based methods present the advantage of being extremely controllable: because they are based on the simulation of a real-life phenomenon, every parameter involved (such as the density of the drops of rain or the number of clapper) can easily be interpreted and manipulated. Although physical simulation have historically been avoided for being computationally heavy, this argument holds little weight in light of progress made in the computational capacities of standard computers. Rather, their common downside lies in their fundamental working: because they are based on a physical models, those synthesis methods are only able to create a single sort (or very similar kinds) of texture. This means that despite their individual performances

regarding realism or controllability, their flexibility is by design extremely poor.

In order for an algorithm to be flexible, it seems to follow that it needs not to work on a given phenomenon but rather on sound texture signals in general.

3.1.2 Granular synthesis

This is the case with granular synthesis. Although they differ in their execution, the base idea of all (re-)synthesis methods presented in this section is the same: to analyze an existing sound texture signal, decompose it into sonic grains and then reassemble it while following a given method in order to create a new sound texture of the same kind.

In [Saint-Arnaud and Popat 1995], those grains are chosen to be the point of a time-frequency representation with eight frequency bands, each being an octave wide. In order to capture the relations between those grains, given intervals in time and frequency are selected in order to form a mask. This mask is then passed along the time-frequency axes and the values of the grains at each of the 14 "holes" of the mask are recorded into a 14-dimension histogram. This histogram is then used to approximate the probability mass function (PMF) of the signal. From there, and starting with the first frame and the first frequency band, this PMF is used to recreate a probable time-frequency representation following the same high-order organization as the original sound texture. This results in an algorithm supposedly capable of synthesizing a wide variety of textures, and rid of the limitations of physics-based synthesis. But by the authors' own admission, the range of textures it manages to create is not particularly wide: due to the roughness of the time-frequency representation, harmonic sounds (such as those including voices) are hard to recreate, while those presenting a periodicity or long-term organizations are poorly synthesized.

In [Lu et al. 2004], the Mel-frequency cepstral coefficients (MFCC, the DFT of the logarithm of the Mel-spectrogram often used in speech recognition) of the original texture signal is first segmented into grains, or sub-clip. This segmentation is performed by gathering together frames that are similar to each others. This similarity is also used to establish the transition probability between grains. During the synthesis process, a series of grain is then selected so as to maximise this probability between each grain and the next. So as to avoid exactly repeating the original signal or endlessly looping over a few grains, the algorithm is prevented from choosing a grain that was close (time-wise) from the previous in the original texture. While it is able to synthesize simple textures, this method fails whenever the high-level organization of the texture is too complex. Due to the MFCC putting less focus on the pitch of the signal, it also performs poorly on textures containing harmonic sounds.

In [Fröjd et al. 2007], a simpler approach is presented. Blocks of varying sizes are randomly taken from the original sound texture waveform to act as grains. To synthesize a texture, those blocks are overlap-added using cross-fades: this means that the amplitude of each block is decreased while that of the following is increased to avoid any discontinuity. Just like [Lu et al. 2004], blocks that are too close to each others in the original are prevented from being neighbors in the synthesis. While this basic process yields satisfying results when synthesizing simple textures with

weak global organization (e.g. rain falling), those results are considerably worse when synthesizing texture with stronger organization (e.g. birds singing).

In [Schwarz et al. 2015], the standard granular synthesis of the kind presented in [Fröjd et al. 2007] is improved by selecting the following block based on its closeness to the current one. This distance is computed using hand-crafted audio descriptors. [Parker et al. 2004] also improves on [Fröjd et al. 2007] by using methods borrowed from visual texture synthesis and presented in [Efros et al. 2001]. After splitting the original texture waveform into equally long segments, those grains are reassembled using overlap-add and cross-fades. To select the following grain, a search through the signal is conducted to identify a region similar to the end of the current one (barring the origin of the grain, to avoid exactly copying the signal) : the block following the similar pattern is chosen as next.

In [O'Leary et al. 2016], textures are described on two levels: low-level atoms and high-level segments. A new texture is synthesized by using the segments of the original as template, and randomly swapping their atoms for others that resemble them. This resemblance is given by the distance between the envelopes of the frequency sub-bands of the two atoms being considered. Segments are then chained together so that the end of one may act as the beginning of the next. This approach aims at conserving the high-level organization of the original texture while simultaneously introducing randomness on a lower scale.

In [Dubnov, Bar-Joseph, et al. 2002], the wavelet transform of the synthesized signal is created by mimicking patterns presents in the transform of the original. The synthesized wavelet transform is progressively created by randomly choosing from the patterns of the original that are similar to its own, and copying the subsequent elements of the wavelet transform. This is done while allowing an error margin when looking for similar patterns: if this margin is too low an exact copy of the original is created, while if it is too high the synthesized texture will be too random and discontinuous. In practice, it appears that many synthesized example present discontinuities. This approach is inspired by the equivalent work performed on visual textures and presented in [Bar-Joseph et al. 2001]. It is then extended and generalized to any sort of representation of the signal in [Dubnov, Assayag, et al. 2007].

Overall, granular methods all present the advantage of being flexible: unlike physics-based methods, they are not dedicated to the synthesis of one single type of texture. This being said, the complexity of the textures they are able to generate still varies from algorithm to algorithm: simpler ones such as [Fröjd et al. 2007] are restricted to textures containing only short-lived high-level organizations. Their main downside lies in their control parameters. While most methods possess a form of control, those parameters are harder to interpret than physical parameters: all we can control are usually similarity criteria or grain lengths.

We can also discern a new problematic that appears when performing texture synthesis via the re-synthesis of an existing texture: it is necessary to be careful when designing the algorithm so as to avoid merely recreating the exact texture. On the contrary, it is also necessary to partially mimic the organization of the original

texture in order to avoid synthesizing a texture that is too random and unlike the original. This equilibrium between copy and randomness for re-synthesis methods also leads to another interrogation: how do we judge if two texture are similar or not ? This interrogation is further developed in the upcoming state of the art in parametric synthesis.

3.1.3 Miscellaneous synthesis methods

But before delving into it, we need to point out that physics-based and granular synthesis are not the only sort of existing non-parametric synthesis. In this section we present two methods that do not quite belong to any of the two : the first using a manual approach, and the second using a filter-based approach.

In [Di Scipio 1999], an empirical method is introduced using Functional Iteration Synthesis (IFS). Starting from an initial real value, the algorithm uses a parametrized non-linear function (such as the sine function) iteratively applied to itself a given number of times to generate a time signal. This signal is then filtered to obtain the synthesized texture. Although this process can potentially generate a wide array of sounds, it needs in practice a manual tuning of its parameters so as to control the output. Because of the chaotic nature of the synthesis, it is impossible to predict the effect of the tuning and there is no guarantee that a given texture can be synthesized.

In [Athineos et al. 2003], the synthesis process is modelized by a gaussian noise filtered both in the time and frequency domain. The coefficients of both time and frequency filters are estimated using Linear Predictive Coding (LPC) from windowed excerpt of the time signal of the original texture. The noise is then first filtered in the frequency domain (which amounts to shaping its temporal envelope) and then filtered in the time-domain (which amounts to shaping its power spectrum). The result of this process is a re-synthesis of the original texture. This algorithm however is not exactly a synthesis algorithm in the sense we have been using until now: its aim is to reproduce as closely as possible the analyzed texture, and not create variations on it.

3.2 Parametric texture synthesis

The paradigm of parametric synthesis is well illustrated in [Hoffman et al. 2006]. In it, the outline of a sound synthesis algorithm working by analysis-synthesis is presented. In the first step of the process, a set of parameters is extracted from the analyzed sound: ideally, this set of parameter (which the authors names "Perceptual Audio Coding") captures all relevant perceptual information of the signal. In the second, an audio output is created by exploring the parameter space and creating a signal which Audio Coding parameters are close to those of the original sound. Although Hoffman et al. mostly aim at synthesizing a signal that is perceptually identical to the original, the aim of a parametric sound texture synthesis is slightly different: in the case of textures, we only want the synthesized signal to belong to the same category of texture as the original and not to be its copy.

This, at its core, is parametric synthesis: finding a set of parameters that captures the audio characteristics of the sounds we wish to synthesize, and then creating a sound signal that possesses the same parameters as a target sound.

Incidentally, the main complexity of parametric synthesis is also exposed in [Hoffman et al. 2006]: the choice of parameters. On this subject, Hoffman et al. only mention that the parameters should be tailored so as to represent the very minimal amount of data needed to store a sound without giving practical examples. This parametrization indeed needs to be sufficiently descriptive: the parameters should hold enough information so as to be able to recreate a sound texture belonging to the same category as the original. This task gets more complex if we want the algorithm to be flexible, since the parametrization should be adapted to a wide array of sounds.

Because the main aim of [Hoffman et al. 2006] is to compress sounds and reproduce them as closely as possible, having an "over-descriptive" parametrization (i.e. storing superfluous information, so long as the necessary one is included) is not too penalizing for them: its only consequence is the worsening of the level of compression. This is not the case when using parametrization for sound texture synthesis. Since we aim for variability in synthesized textures, we need to avoid having the output of the algorithm be an exact copy of the original one. This means that the parametrization needs to be general enough so that it does not contain precise information allowing the exact reproduction of the original, but rather broader information allowing the construction of a texture belonging to the same category.

This reveals a problematic that is crucial in parametric synthesis, and which echoes the inherent issue of re-synthesis methods we evoked earlier with granular synthesis. Too strong a parametrization would create identical outputs, while one too weak would result in random outputs: he synthesis algorithm thus needs to reach a balance between the two.

It can also be noted that this search for the right parametrization of textures is equivalent to the search of a perceptual distance for textures. From a parametrization, one could build a distance that is simply the euclidian distance between parameters and that reaches zero once two sounds belong to the same category of textures. From a perceptual distance, one would simply need to create a sound that is at a zero distance from the original texture to generate a texture of the same category (in which case the parametrization is implicitly contained in the distance).

Having introduced parametric texture synthesis, we may now give an overview of existing methods. So far, works in this kind of synthesis have been mostly centered around two approaches: one using parametrization based on perceptual statistics, while the others uses CNN-based statistics.

3.2.1 Perceptual-based parametrization

The description of sounds via their statistics is already present in [Attias et al. 1997]. In this work, the perceptual spectrograms of different recordings of natural sounds (such as animal and environmental recordings, but also speech and music) are studied in order to highlight similarities between recordings of a same category. Notably, the

distributions of values in each bands as well as their Fourier transforms seem to vary little within the same category.

This idea is developed further and applied exclusively to sound textures in [McDermott, Schemitsch, et al. 2013]. Because sound textures contain an important number of individual events, the authors argue that it is very unlikely that our brains store the entirety of this information. This could imply that our brains use a more compact representation of textures, such as statistics. In order to emphasize this point, two experiments are performed. In a first experiment, subjects need to judge whether the texture they are hearing are perfectly identical or simply part of the same recording. In this case, performance decreases with excerpt length. In a second, subjects need to judge whether the excerpts they are hearing are part of the same recording or not. Here performance increases with excerpt length. The results of the first support the idea that while excerpts are short we are able identify salient details that we use to discriminate them. Once they outgrow the length of about a hundred milliseconds, this ability keeps worsening as remembering salient events becomes increasingly difficult. Despite that, and as shown with the results of the second, it would seem that our ability to tell whether two textures are of the same kind is at its best when the excerpts are as long as possible. Since we are not remembering salient events at this scale, this supports the idea that we use a summary of the organization of the atoms, such as statistics, to identify textures.

Those results serve as an *a posteriori* justification to the sound texture synthesis algorithm previously introduced in [McDermott and Simoncelli 2011]. Inspired by the similar work in visual perception of [Julesz 1962] and [Portilla et al. 2000], this algorithm indeed uses the statistics of (and between) the frequency sub-bands of a sound signal as parameters for its parametric synthesis. It is important to note that in this work the synthesis is not a goal by itself, but rather a mean to explore which statistics are computed by the human brain in order to differentiate textures. This is notably the reason why the frequency sub-bands used in this methods are designed to approximate the working of the human cochlear filtering.

This work is extended in [Liao 2015], which propose the use of another sound representation and statistics in order to fasten the synthesis computation time, as well as in [Kim et al. n.d.], which propose a more complete parametrization of textures in order to increase the quality of the synthesis.

3.2.2 CNN-based parametrization

Appearing with [L. Gatys et al. 2015] in which it is used for visual texture synthesis, the other major parametrization method is CNN-based. In this work, the VGG-19 CNN (a network trained in image recognition) is used in a slightly indirect way: instead of re-training it or using it for prediction, the authors use it to extract a set of parameters from its input. Those parameters are set as the inter-correlation between the feature maps of the network. The results of the parametric synthesis using this set of parameters are convincing, and the algorithm can be used on a wide range of visual textures. This method is also successfully adapted for style transfer (i.e. the process of giving to an image the artistic style of another) in [L. A. Gatys et

al. 2016].

Inspired by the success of this synthesis in the visual domain, several attempts have been made to adapt it to the audio domain. This thesis includes such an attempt, but other works have pursued the same goal around the same time as we have.

In [Ulyanov et al. n.d.], the CNN-based parametrization is directly applied to sound texture spectrograms: this results in a synthesized image that is then considered as a spectrogram itself, and inverted using the Grinffin-Lim algorithm (mentioned in Section 2.1.2) in order to retrieve an audio signal. This method is only mildly successful seeing as the synthesized textures contain an important quantity of artefacts, potentially due to the spectrogram inversion process. In addition to this, the showcased examples mostly contain very rhythmic sounds (e.g. fireworks, fusillades) or sounds poorly matching our definition of sound texture (e.g. single English-speaking voice, music).

In [Antognini et al. 2018], the same method is developed further by slightly modifying the usual process of parametric synthesis. To the usual parametric loss that is minimized throughout the synthesis process, the authors add two others. The first is an auto-correlation loss that is used in order for the synthesis to better reproduce rhythmic organizations from the original texture. The second is a loss designed so that the synthesized sound is as different as possible from the original, compensating the over-descriptive side of the other two. As with [Ulyanov et al. n.d.], the algorithm yields a spectrogram that is inverted using the Griffin-Lim algorithm. Textures synthesized by this algorithm are more realistic than those presented in [Ulyanov et al. n.d.], although artefacts are still present. Additionally, the flexibility of the synthesis is hurt by the fact that the three losses may need to be tuned manually for each texture.

The paradigm of CNN-based parametrization has also been adapted to audio style transfer, notably in [Grinstein et al. 2018], [Tomczak et al. 2018] and [Barry et al. 2018]. But the concept of audio style transfer being at least as nebulous and ill-defined as that of sound texture, we leave this task to others and are content with merely mentioning the existence of those works.

It must also be noted that since our work heavily revolves around parametric sound texture synthesis, both statistics-based and CNN-based parametrization are detailed and discussed in greater length in the following chapters.

References for chapter 3

Antognini, J., Hoffman, M., and Weiss, R. J. (2018). "Synthesizing diverse, high-quality audio textures". In: *arXiv preprint arXiv:1806.08002* (cit. on pp. 43, 84, 85, 88, 92, 96, 100, 106–108, 112, 123).

Athineos, M. and Ellis, D. P. (2003). "Sound texture modelling with linear prediction in both time and frequency domains". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cit. on pp. 2, 40).

Attias, H. and Schreiner, C. E. (1997). "Temporal low-order statistics of natural sounds". In: *Advances in neural information processing systems* (cit. on p. 41).

Bar-Joseph, Z. et al. (2001). "Texture mixing and texture movie synthesis using statistical learning". In: *IEEE Transactions on visualization and computer graphics* (cit. on p. 39).

Barry, S. and Kim, Y. (2018). *"Style" Transfer for Musical Audio Using Multiple Time-Frequency Representations* (cit. on pp. 43, 82).

Chadwick, J. N. and James, D. L. (2011). "Animating fire with sound". In: *ACM Transactions on Graphics (TOG)* (cit. on p. 37).

Di Scipio, A. (1999). "Synthesis of environmental sound textures by iterated nonlinear functions". In: *Proceedings of the 2nd COST G-6 Workshop on Digital Audio Effects (DAFx)* (cit. on p. 40).

Dubnov, S., Assayag, G., and Cont, A. (2007). "Audio oracle: A new algorithm for fast learning of audio structures". In: (cit. on p. 39).

Dubnov, S., Bar-Joseph, Z., et al. (2002). "Synthesizing sound textures through wavelet tree learning". In: *IEEE Computer Graphics and Applications* (cit. on p. 39).

Efros, A. A. and Freeman, W. T. (2001). "Image quilting for texture synthesis and transfer". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (cit. on p. 39).

Fröjd, M. and Horner, A. (2007). "Fast sound Texture synthesis using Overlap-Add." In: *ICMC* (cit. on pp. 38, 39).

Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). "Image style transfer using convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (cit. on pp. 42, 70, 124).

Gatys, L., Ecker, A. S., and Bethge, M. (2015). "Texture synthesis using convolutional neural networks". In: *Advances in neural information processing systems* (cit. on pp. 42, 48, 70, 72–74, 76, 79, 82, 83, 90, 92, 101, 123, 125).

Grinstein, E. et al. (2018). "Audio style transfer". In: *2018 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2018. Proceedings.(ICASSP'18)*. (Cit. on p. 43).

Hoffman, M. D. and Cook, P. R. (2006). "Feature-Based Synthesis: Mapping Acoustic and Perceptual Features onto Synthesis Parameters." In: *ICMC* (cit. on pp. 40, 41).

Julesz, B. (1962). "Visual pattern discrimination". In: *IRE transactions on Information Theory* (cit. on p. 42).

Kim, H.-S. and Smith, J. (n.d.). "Synthesis of sound textures with tonal components using summary statistics and and all-pole residual modeling". In: (cit. on pp. 42, 66).

Liao, W.-H. (2015). "Modelling and transformation of sound textures and environmental sounds". PhD thesis (cit. on pp. 42, 48).

Lu, L., Wenyin, L., and Zhang, H.-J. (2004). "Audio textures: Theory and applications". In: *IEEE transactions on speech and audio processing* (cit. on pp. 6, 38).

McDermott, J. H., Schemitsch, M., and Simoncelli, E. P. (2013). "Summary statistics in auditory perception". In: *Nature neuroscience* (cit. on pp. 42, 48).

McDermott, J. H. and Simoncelli, E. P. (2011). "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis". In: *Neuron* (cit. on pp. 42, 48, 49, 52, 54, 55, 60, 62, 63, 65, 71, 82, 85, 88, 107, 108, 112, 115, 122, 123).

Miklavcic, S. J., Zita, A., and Arvidsson, P. (2004). *Computational real-time sound synthesis of rain*. Department of Science and Technology (ITN), Campus Norrköping, Linköping ... (cit. on p. 36).

Moss, W. et al. (2010). "Sounding liquids: Automatic sound synthesis from fluid simulation". In: *ACM Transactions on Graphics (TOG)* (cit. on p. 37).

Oksanen, S., Parker, J., and Välimäki, V. (2013). "Physically informed synthesis of jackhammer tool impact sounds". In: *Digital Audio Effects (DAFx)* (cit. on p. 37).

O'Leary, S. and Röbel, A. (2016). "A montage approach to sound texture synthesis". In: *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* (cit. on p. 39).

Parker, J. R. and Behm, B. (2004). "Creating audio textures by example: tiling and stitching". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing* (cit. on p. 39).

Peltola, L. et al. (2007). "Synthesis of hand clapping sounds". In: *IEEE Transactions on Audio, Speech, and Language Processing* (cit. on p. 37).

Portilla, J. and Simoncelli, E. P. (2000). "A parametric texture model based on joint statistics of complex wavelet coefficients". In: *International journal of computer vision* (cit. on pp. 42, 50, 70, 71, 73, 123).

Saint-Arnaud, N. and Popat, K. (1995). "Analysis and synthesis of sound textures". In: *in Readings in Computational Auditory Scene Analysis* (cit. on pp. 4, 38).

Schwarz, D. and O'Leary, S. (2015). "Smooth granular sound texture synthesis by control of timbral similarity". In: (cit. on p. 39).

Tomczak, M., Southall, C., and Hockman, J. (2018). "Audio Style Transfer with Rhythmic Constraints". In: *Digital Audio Effects (DAFx)* (cit. on pp. 43, 82).

Ulyanov, D. and Lebedev, V. (n.d.). *Audio texture synthesis and style transfer* (cit. on pp. 43, 83–85, 88, 91, 92, 107, 108, 112, 123).

Chapter 4

Synthesis based on perceptual statistics

Chapter overview

The parametric synthesis method investigated in this chapter consists in imposing onto a base signal the perceptual statistics extracted from an original texture, in order to create a new texture of the same kind. Our contributions to it are an improvement and simplification of the imposition process using Wirtinger calculus, a modification of the perceptual filter-banks in favor of minimum-phase filter-banks and an investigation of the base signal initialization.

Contents

The state of the art in sound texture synthesis gives us a context in which we can inscribe our work. Out of existing methods, all come with both advantages and drawbacks: physics-based methods are controllable but not flexible in the least, granular methods are flexible but poorly controllable, and parametric methods are potentially extremely flexible and realistic, but totally dependent on the quality of their parametrization.

Our work focuses exclusively on parametric synthesis, and initially started as an extension of the algorithm presented in [McDermott and Simoncelli 2011]. The following section details our reasons for opting for this algorithm.

4.1 Motivation

The first reason for working with the parametric synthesis introduced in [McDermott and Simoncelli 2011] was very practical: our work followed the PhD of Wei-Hsiang Liao ([Liao 2015]), a former member of the Analysis/Synthesis team at IRCAM, which was centered around this method. This means that both expertise and codes were readily available within the team. Several other reasons also motivated us toward trying to build on this existing method.

Out of those, the main one was the quality and flexibility of the algorithm: although its authors claim that their primary goal when designing it was not to recreate sounds as realistically as possible, the results of the perceptually-based parametric synthesis easily matched that of others state of the art algorithms at the time it was published. In addition to this, the algorithm worked efficiently for a wide array of sounds, ranging from recordings of crickets to hubbubs of crowds.

The fact that the algorithm was designed to mimic the perceptual processing of sound by the human ear, notably by using perceptually-based frequency sub-bands computation, was also particularly interesting. As evoked in [McDermott, Schemitsch, et al. 2013] and in [McWalter et al. 2013], this implies that a successful parametrization of sound textures could serve as an insight to the inner working of the human auditory system. Outside of that, and since the texture synthesis algorithm presented in [L. Gatys et al. 2015] had not yet gained traction in the audio field around the beginning of this work, this method was also attractively innovative.

This being said, this sound texture synthesis method had room for improvement: notably, several aspects of the algorithm were harmful to the realism and flexibility of the synthesis. This, combined to the previous reasons, is why we initially decided to work with this perceptual paradigm of texture synthesis.

In order to be able to detail precisely the working and the weaknesses of this algorithm, and so as to lay the ground for our improvement attempts, we start with giving a more thorough explanation of it than what was given in the state of the art.

4.2 Explanation of the algorithm

As is the case with most re-synthesis algorithms, the perceptual sound texture synthesis method introduced in [McDermott and Simoncelli 2011] can be broken down into two steps. In the first, the signal is analyzed to extract a set of statistics, acting as parameters to the parametric synthesis, while in the second those parameters are imposed onto an existing signal to generate a new texture.

4.2.1 Analysis

The analysis itself can also be broken down into two steps: the extraction of the frequency sub-bands of the signal, followed by the computation of its statistics.

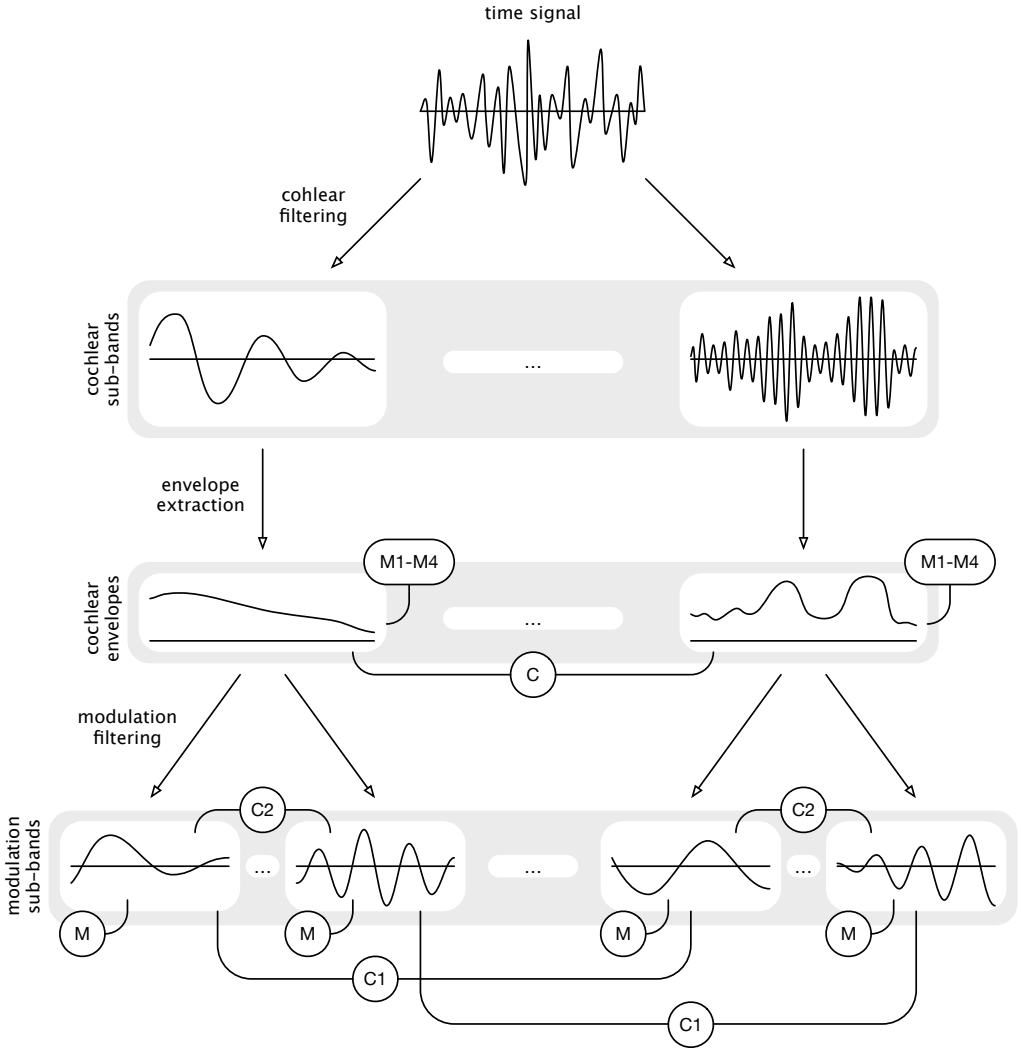


Figure 4.1: *Extraction of the frequency sub-bands and their statistic. The time signal is passed through a bank of cochlear filters to give the cochlear sub-bands. The envelope of those sub-bands is computed and compressed to give the cochlear envelopes, which are then passed through the modulation filter-bank to give the modulation sub-bands. Moments $M1$, $M2$, $M3$, $M4$ and cross-correlations C are computed from the cochlear envelopes. Moments M and cross-correlations $C1$ and $C2$ are computed from the modulation sub-bands.*

Sub-band extraction

In order to approximate the working of the first layers of the human auditory system, the sound signal is passed through a cascade of two filter banks. The first is a bank of band-pass filters linearly spaced on the ERB scaled (c.f. Section 2.1.3), named cochlear filters. In the original paper, 30 of those filters span 52 to 8848 Hz. Those filters are also designed to equally cover the frequency axis: this means that the sum of those filters acts as the identity functions for frequencies contained within the range of the bank. As long as the initial signal does not possess any frequency content outside of this range, it can thus be reconstructed by simply summing the outputs of the filter bank.

The envelopes of those sub-bands are then computed using the analytic extension of the outputs of the filter. Given a real-valued signal x and its DFT X of length N, its complex-valued analytic extension x_a can be defined as:

$$x_a = DFT^{-1}(X_a) \quad \text{with} \quad X_a(k) = \begin{cases} X(k), & \text{if } k = 0 \text{ or } k = \frac{N}{2} \\ 2X(k), & \text{if } k \in [1, \frac{N}{2} - 1] \\ 0, & \text{if } k \in [\frac{N}{2} + 1, N - 1] \end{cases} \tag{4.1}$$

The magnitudes of those analytic signals are the instantaneous envelope of the sub-bands. These envelopes are then used to divide their respective (real-valued) sub-bands in order to obtain the "fine-structure" signals of each subs-bands. Afterwards, the envelopes are compressed (by raising them to the power of 0.3) in order to imitate the working of the cochlea. Those compressed envelopes are referred to as cochlear envelopes.

Staying consistent with existing audition models, each of those envelopes is then processed by a second filter bank, named modulation filters. Similarly to the cochlear filters, the modulations filters are band-pass filters and are linearly spaced on a logarithmic scale. In the original paper, 20 of those filters span 0.5 to 200 Hz. The fact that this frequency range is an order of magnitude lower than that of the cochlear filters is due to the fact that envelopes vary much more slowly than the original audio signal. It also allows the envelopes to be down-sampled (in this case, to a sampling frequency of 400 Hz) without harm so as to lighten computations. The outputs of those filters are referred to as the modulation bands of each envelope.

The sub-band extraction process is summed up on Figure 4.1.

Statistics extraction

The statistics used as parameters for this method are also chosen based on perceptual motivations. Using cochlear envelopes and modulations bands as a perceptual representation of the signal, the goal of this parametrization is to select statistics that are both plausibly measured in the neural processing of sound (i.e. using basic operations such as squaring and products) and that vary significantly between different textures. The choice of statistics is also inspired by older and similar works performed in the visual domain, such as [Heeger et al. 1995] and [Portilla et al. 2000].

So as to avoid any artefacts caused by the assumed circular boundary conditions of the DFT, a window w spanning the whole signals is used. This window reaches zero at both ends of the signals, and verifies $\sum_n w(n) = 1$. n refers to a time index, and we denote by s_k the k^{th} cochlear envelope, and its p^{th} modulation band by $b_{k,p}$. Using those notations, the chosen statistics are:

- the mean (or first moment) $M1$ of each cochlear envelope:

$$M1_k = \mu_k = \sum_n w(n) s_k(n) \tag{4.2}$$

- the normalized variance (or second moment) $M2$ of each cochlear envelope:

$$M2_k = \frac{\sigma_k^2}{\mu_k^2} = \frac{\sum_n w(n)(s_k(n) - \mu_k)^2}{\mu_k^2} \tag{4.3}$$

- the normalized skewness (or third moment) $M3$ of each cochlear envelope:

$$M3_k = \frac{\sum_n w(n)(s_k(n) - \mu_k)^3}{\sigma_k^3} \tag{4.4}$$

- the normalized kurtosis (or fourth moment) $M4$ of each cochlear envelope:

$$M4_k = \frac{\sum_n w(n)(s_k(n) - \mu_k)^4}{\sigma_k^4} \tag{4.5}$$

- the cochlear cross-band envelope correlation C between pairs of cochlear envelopes:

$$C_{kl} = \sum_n \frac{w(n)(s_k(n) - \mu_k)(s_l(n) - \mu_l)}{\sigma_k \sigma_l} \tag{4.6}$$

- the modulation power M of each modulation band:

$$M_{k,p} = \frac{\sum_n w(n)(b_{k,p}(n)^2}{\sigma_k^2} \tag{4.7}$$

- a first modulation correlation $C1$ between modulation bands of different envelopes, but tuned to the same modulation frequency:

$$C1_{kl,p} = \frac{\sum_n w(n) b_{k,p}(n) b_{l,p}(n)}{\sigma_{k,p} \sigma_{l,p}} \tag{4.8}$$

  with:

$$\sigma_{k,p} = \sqrt{\sum_n w(n) b_{k,p}(n)^2} \tag{4.9}$$

- a second modulation correlation $C2$ between modulation bands of the same envelope, but which modulation frequency are spaced by an octave (here $q > p$):

$$C2_{k,pq} = \frac{\sum_n w(n) \overline{d_{k,p}(n)} a_{k,q}(n)}{\sigma_{k,p} \sigma_{k,q}} \tag{4.10}$$

  with $a_{k,p}$ the analytic extension of $b_{k,p}$ and:

$$d_{k,p}(n) = \frac{a_{k,p}^2}{|a_{k,p}|} \tag{4.11}$$

This set of statistics taken together constitute the parametrization of textures proposed in [McDermott and Simoncelli 2011]. It is also important to note that while McDermott et al. are confident that the brain uses a statistical summary of texture in order to discriminate them, they do not claim that those statistics are identical to the statistics that the brain computes: they are mostly used as examples to prove the validity of this texture parametrization.

4.2.2 Synthesis

Having extracted this set of statistics from an existing texture, the following part of the synthesis algorithm is to impose them onto an existing sound. This existing sound will henceforth be referred to as the base signal, in contrast to the existing texture referred to as the original or target signal.

The imposition proposed by McDermott et al. is slightly convoluted, and mainly consists in progressively modifying the cochlear envelopes of the base sound while keeping its fine-structure signals intact. To do so, the algorithms first starts with the cochlear sub-bands with the highest power. An objective function is then set as the squared error between the statistics of the original signal and the base signal for this specific sub-band. Using a variant of the gradient descent algorithm described in Section 2.3.4, the cochlear envelope of the base signal is iteratively modified until the objective function reaches a given threshold. From there the sub-band is reassembled by multiplying the new envelope with the existing fine-structure signal, and the base signal is reassembled by adding together the cochlear sub-bands (including the recently modified one). This process is then repeated for all sub-bands so as to modify the base signal until it statistics are all close to those of the original signal. Figure 4.2 illustrates this imposition algorithm.

In the case of cross-correlations across cochlear sub-bands (such as C and $C1$), those statistics are only taken into account within the objective function of the last of the two sub-bands to be modified. This is not optimal, since it would be better to also include those statistics within the objective function of the first sub-band of each couple, but this flaw is inherent to the fact that the sub-bands are modified one after the other. In addition to this, the modified sub-bands are not necessarily consistent with each others in the sense that the signal obtained by re-combining them, once decomposed, does not always yield the same sub-bands. This is similar to the notion of consistency in STFT matrices (mentioned in Section 2.1.2): it is possible that the new sub-bands created by modifying their old envelope now contain frequencies that were not initially part of this sub-band. This also implies that sub-bands onto which statistics imposition have already been performed can be modified when imposing statistics on the next sub-bands. The fact that sub-bands with high power are prioritized and modified first is meant as a way to minimize this issue.

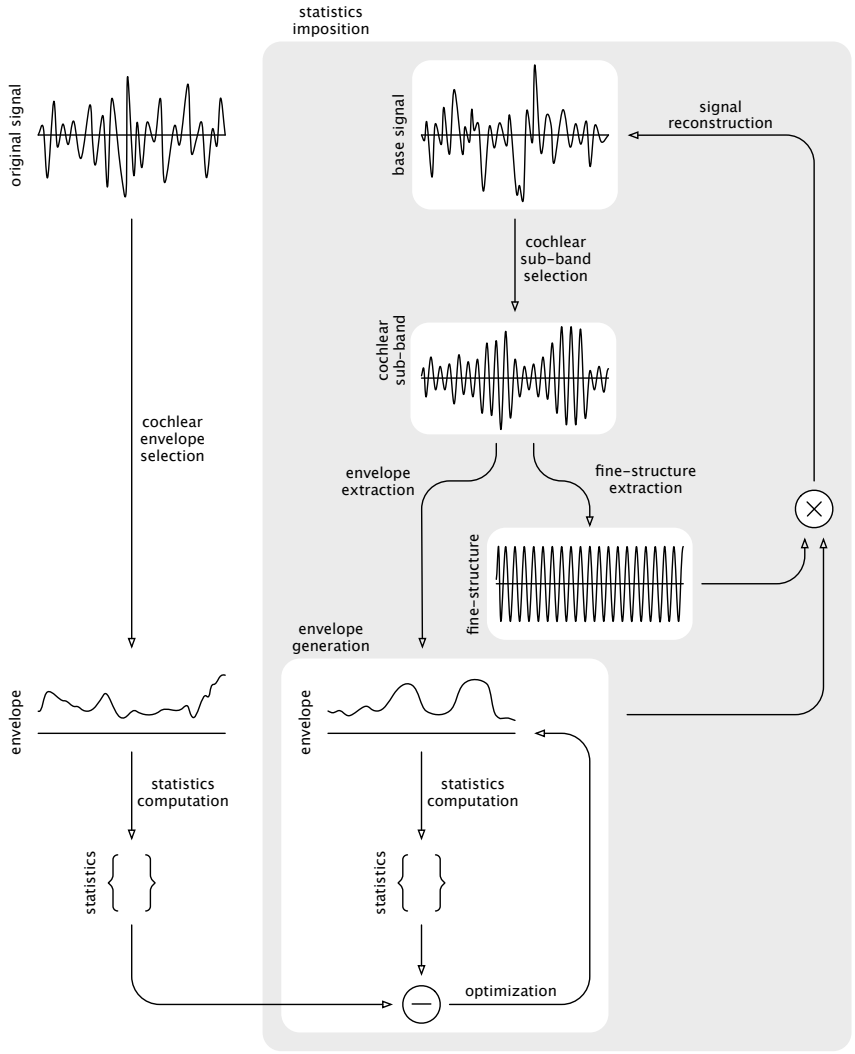


Figure 4.2: *Imposition of the statistics from the original signal onto a base signal. The envelope of each cochlear sub-band is extracted and optimized so that its statistics are similar to those of the original sound. The envelope is reassembled to the fine-structure of the sub-band and the base signal reconstructed, before performing the same operations on the next cochlear sub-band and until all are modified.*

4.2.3 Results

Audio results for this method are available online at `http://mcdermottlab.mit.edu/texture_examples/index.html`.

While this algorithm works well on most presented textures, its results are best when the target texture is stationary: recognizable salient patterns such as the insects chirps of the "insects" texture (on the second example page) than span over a longer duration give poorer results. From our testings, the algorithm also struggles when the target texture presents salient events that are rarely repeated: because those events can hardly be described statistically, they end up hurting the statistics of the texture atoms and are chopped and distorted on synthesis. This could be interpreted as the algorithm working with the stronger, more restrictive definition of sound texture presented in Section 1.1.2. As mentioned in the discussion of the definition of sound textures, this results in discarding an important part of what are commonly called

texture, as well as most real-life recordings of textures that often contain un-textural distinctive events.

In addition to this, all textures presenting sharp impacts (such as the cracks in the texture "fire") are poorly recreated in the synthesized sound, and result in audible chirping artefacts. Textures containing tonal sounds (such as howling of the wind in the texture "wind") are also not well synthesized. This would tend to indicate that the algorithm struggles at imposing long correlations across frequency bands, or throughout time.

This concludes our detailed explanation of the sound texture synthesis method presented in [McDermott and Simoncelli 2011]: for the remainder of this chapter, the work presented is solely ours.

4.3 Time domain imposition: Wirtinger calculus

Out of the weak points of the method presented in the previous Section, we first focused on establishing a better way of imposing statistics on the cochlear sub-bands than iteratively altering their envelopes. Indeed, this sub-optimal imposition method directly impacts the results of the synthesis algorithm, independently of the statistics used, and thus seemed like the most fundamental flaw of the method.

Our proposition to solve the leakage occurring when the imposition on a band harms the previous ones and to avoid taking correlations into account only during the imposition on the second sub-band is as follow: to perform all impositions at once, directly on the base time signal instead of on its sub-bands.

4.3.1 Mathematical notations

So as to be able to clearly detail the troubles that this time imposition entails, let us first explicitly detail the mathematical notations used in this Section.

For a differentiable function $f\colon \mathbb{R} \to \mathbb{C}$, its real derivative at $a \in \mathbb{R}$ is denoted:

$$\frac{\partial f}{\partial x}(a) \tag{4.12}$$

By extension, the real gradient at $a \in \mathbb{R}^M$ of a differentiable and multi-variable function $f\colon \mathbb{R}^M \to \mathbb{C}$, is also denoted:

$$\frac{\partial f}{\partial x}(a) = \left(\frac{\partial f}{\partial x_1}(a), \frac{\partial f}{\partial x_2}(a), ..., \frac{\partial f}{\partial x_M}(a)\right) \tag{4.13}$$

Any complex number $c \in \mathbb{C}$ can be decomposed as $a + ib$ with $(a, b) \in \mathbb{R}^2$ its real and imaginary parts (also denoted by $\mathrm{Re}(c)$ and $\mathrm{Im}(c)$). Similarly, any function $f\colon \mathbb{C} \to \mathbb{C}$ can be considered as a function of $\mathbb{R}^2 \to \mathbb{C}$ with $f(c) = f(a, b)$. If it exists, the derivative of f at c with respect to the real part of its input is denoted by:

$$\frac{\partial f}{\partial x}(c) \tag{4.14}$$

This matches our previous notations, since this derivative is also the real derivative of f when its domain is restrained to $\mathbb{R}$. If it exists, the derivative of f at c with respect to the imaginary part of its input is denoted by:

$$\frac{\partial f}{\partial y}(c) \tag{4.15}$$

4.3.2 Difficulties of time imposition

Previously, a cochlear objective function was expressed as the distance between the statistics of the base signal and those of the original signal for a particular cochlear envelope. These statistics are those of the envelope, meaning moments $M1-4$ and potentially correlations C, but also those of the modulation sub-bands of the envelope, meaning moments M as well as correlations $C2$ and potentially correlations $C1$. In order to perform an optimization on the envelope, most methods (such as the gradient descent) require the explicit gradient of the objective function with respect to the signal to be optimized. In the current case, said signal is the cochlear envelope that is being imposed on.

In order to perform the imposition directly on the base time signal, we first regroup all distances between the statistics of the original and base signals and for all sub-bands in the same objective function. The following step is less straightforward, and requires being able to express the gradient of this function with regard to the base time signal. The main hurdle in doing so (and possibly the reason why time imposition is not used in [McDermott and Simoncelli 2011]) is the fact that the computation of the sub-bands and their envelopes involves the frequency domain, which is complex-valued. Indeed, the cochlear sub-bands can be obtained by filtering the spectrum of the time signal while their envelopes are computed using the Hilbert transform, which can also be interpreted as a multiplication in the frequency domain.

When the objective function is the result of the composition of a succession of differentiable real-valued functions, the gradient of the objective function can be obtained using the usual chain rule with the gradients and jacobians of each individual function. In the case of our time imposition, some of the functions composing the objective function are either complex-valued or use complex parameters. Since the base time signal is a real-valued signal and the objective function (being a distance) is also real-valued, this could be illustrated as having the objective function $\mathcal{E}$ be the result of the composition of two functions: $f\colon \mathbb{C} \to \mathbb{R}$ and $g\colon \mathbb{R}^N \to \mathbb{C}$, with N the length of the base time signal. While it would be tempting to simply switch to using the $\mathbb{C}$-differentiability, it is easy to demonstrate that any real-valued function with complex parameters is not $\mathbb{C}$-differentiable. Indeed, so as to be $\mathbb{C}$-differentiable f would need to fulfill the Cauchy-Riemann conditions expressed as:

$$\frac{\partial \operatorname{Re}(f)}{\partial x} = \frac{\partial \operatorname{Im}(f)}{\partial y} \tag{4.16}$$

$$\frac{\partial \operatorname{Re}(f)}{\partial y} = -\frac{\partial \operatorname{Im}(f)}{\partial x} \tag{4.17}$$

Because the output of f is real, $\operatorname{Im}(f)$ is null which means f needs to be a constant function to be $\mathbb{C}$-differentiable. This case being out of question, using $\mathbb{C}$-differentiability is thus not an option.

4.3.3 Introduction to Wirtinger calculus

Despite that, if f is differentiable with respect to both the real and imaginary part of its input (a property we henceforth designate as being differentiable in the real sense) it is always possible to interpret it as a differentiable function of $\mathbb{R}^2 \to \mathbb{C}$ by splitting its inputs into their real and imaginary parts. This means that differentiability in the real sense is sufficient when looking to optimize f, despite this property being weaker than $\mathbb{C}$-differentiability since it does not fulfill the Cauchy-Riemann conditions.

This is where Wirtinger calculus intervenes. Introduced in [Wirtinger 1927], it is a way of manipulating both partial derivatives of a function of $\mathbb{C} \to \mathbb{C}$ that is only differentiable in the real sense without going through the trouble of treating them individually. In addition to this, Wirtinger calculus acts as a bridge toward $\mathbb{C}$-differentiability since it overlaps with $\mathbb{C}$-differentiability when the complex derivative exists.

For $f\colon \mathbb{C} \to \mathbb{C}$ differentiable in the real sense, its Wirtinger derivative (henceforth W-derivative) is denoted by and defined as:

$$\frac{\partial f}{\partial z} = \frac{1}{2}\left(\frac{\partial f}{\partial x} - i\frac{\partial f}{\partial y}\right) \tag{4.18}$$

While its conjugate Wirtinger derivative (henceforth $\overline{W}$-derivative) is denoted by and defined as:

$$\frac{\partial f}{\partial \overline{z}} = \frac{1}{2}\left(\frac{\partial f}{\partial x} + i\frac{\partial f}{\partial y}\right) \tag{4.19}$$

Manipulating those two derivatives is equivalent to manipulating the two real partial derivatives since we can easily switch from one expression to the other using:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial z} + \frac{\partial f}{\partial \overline{z}} \tag{4.20}$$

$$\frac{\partial f}{\partial y} = i\left(\frac{\partial f}{\partial z} - \frac{\partial f}{\partial \overline{z}}\right) \tag{4.21}$$

Since the partial derivatives are enough to optimize a function, so are the Wirtinger derivatives.

This can then be extended to functions of several variables: if f denotes a function of $\mathbb{C}^M \to \mathbb{C}$ and is differentiable in the real sense, meaning here partly differentiable with respect to the real and imaginary parts of all of its inputs, we define the W and $\overline{W}$-gradients of f similarly to their real counterpart:

$$\frac{\partial f}{\partial z} = \left(\frac{\partial f}{\partial z_1}, \frac{\partial f}{\partial z_2}, ..., \frac{\partial f}{\partial z_M}\right) \tag{4.22}$$

$$\frac{\partial f}{\partial \overline{z}} = \left(\frac{\partial f}{\partial \overline{z_1}}, \frac{\partial f}{\partial \overline{z_2}}, ..., \frac{\partial f}{\partial \overline{z_M}}\right) \tag{4.23}$$

As detailed in [Bouboulis 2010], and in the case of a real-valued function (such as our objective function) knowing either the W or $\overline{W}$-gradient of the function is sufficient to minimize it.

4.3.4 Properties of the Wirtinger derivatives

Before applying Wirtinger calculus to our case, let us first detail a few of its more useful properties (more in-depth analysis of this calculus can be found in [Bouboulis 2010; Brandwood 1983; Fischer 2005]):

Link with $\mathbb{C}$-differentiability

If f denotes a function of $\mathbb{C}^{\mathbb{M}} \to \mathbb{C}$ the following property holds:

$$f \text{ is } \mathbb{C}\text{-differentiable} \iff \begin{cases} f \text{ is differentiable in the real sense} \\ \frac{\partial f}{\partial \overline{z}} = 0 \end{cases} \tag{4.24}$$

In the case where f is $\mathbb{C}$-differentiable, both its complex and W-gradients are equal, making Wirtinger calculus a proper extension of $\mathbb{C}$-differentiability.

Linearity

The W and $\overline{W}$-derivations are both linear, meaning for f and g two functions of $\mathbb{C}^M \to \mathbb{C}$ differentiable in the real sense and for $(\alpha, \beta) \in \mathbb{C}^2$:

$$\frac{\partial(\alpha f + \beta g)}{\partial z} = \alpha \frac{\partial f}{\partial z} + \beta \frac{\partial g}{\partial z} \tag{4.25}$$

$$\frac{\partial(\alpha f + \beta g)}{\partial \overline{z}} = \alpha \frac{\partial f}{\partial \overline{z}} + \beta \frac{\partial g}{\partial \overline{z}} \tag{4.26}$$

Function composition

For f and g two functions of $\mathbb{C} \to \mathbb{C}$ differentiable in the real sense, the Wirtinger chain rule gives:

$$\frac{\partial f \circ g}{\partial z} = \left(\frac{\partial f}{\partial z} \circ g\right) \frac{\partial g}{\partial z} + \left(\frac{\partial f}{\partial \overline{z}} \circ g\right) \frac{\partial \overline{g}}{\partial z} \tag{4.27}$$

$$\frac{\partial f \circ g}{\partial \overline{z}} = \left(\frac{\partial f}{\partial z} \circ g\right) \frac{\partial g}{\partial \overline{z}} + \left(\frac{\partial f}{\partial \overline{z}} \circ g\right) \frac{\partial \overline{g}}{\partial \overline{z}} \tag{4.28}$$

This can be extended to the case of functions of several variables: if this time f denotes a function of $\mathbb{C}^M \to \mathbb{C}$ and g denotes a function of $\mathbb{C}^N \to \mathbb{C}^M$, both being differentiable in the real sense, for $n \in [1, N]$ the chain rule gives:

$$\frac{\partial f \circ g}{\partial z_n} = \sum_{m \in [1,M]} \left(\frac{\partial f}{\partial z_m} \circ g\right) \frac{\partial g_m}{\partial z_n} + \left(\frac{\partial f}{\partial \overline{z_m}} \circ g\right) \frac{\partial \overline{g_m}}{\partial z_n} \tag{4.29}$$

$$\frac{\partial f \circ g}{\partial \overline{z_n}} = \sum_{m \in [1,M]} \left(\frac{\partial f}{\partial z_m} \circ g\right) \frac{\partial g_m}{\partial \overline{z_n}} + \left(\frac{\partial f}{\partial \overline{z_m}} \circ g\right) \frac{\partial \overline{g_m}}{\partial \overline{z_n}} \tag{4.30}$$

Complex conjugate

If f denotes a function differentiable in the real sense, the following property holds:

$$\overline{\left(\frac{\partial f}{\partial z}\right)} = \frac{\partial \overline{f}}{\partial \overline{z}} \tag{4.31}$$

In the case where f is real-valued, its derivatives with respect to the real and imaginary parts of its inputs are both real, and such we have:

$$\overline{\left(\frac{\partial f}{\partial z}\right)} = \frac{\partial f}{\partial \overline{z}} \tag{4.32}$$

Meaning that in the case of functions with real-valued output, it is strictly equivalent to manipulate the W and the $\overline{W}$-gradients.

4.3.5 Gradient conversion between time and frequency domain

Since the most frequent source of complex-valued signals in sound signal processing (and the reason for our use of Wirtinger calculus in the first place) is the Fourier transform, we first investigate its effects on the gradients of a given function: in other words, we wish to establish the relationship between the gradient of an objective function with respect to a time signal, and the gradient of the same function with respect to the DFT of this signal.

To express it formally, let us suppose an objective function $\mathcal{E}$, differentiable in the real sense and defined as:

$$\begin{aligned} \mathcal{E}\colon \mathbb{C}^N &\to \mathbb{R} \\ z &\mapsto \mathcal{E}(z) \end{aligned} \tag{4.33}$$

So as to be able to convert a time gradient into a frequency gradient, we suppose that the the value of the W-gradient of $\mathcal{E}$ is known at a given point $c \in \mathbb{C}^N$. Our goal is now to evaluate this gradient at $C \in \mathbb{C}^K$, the DFT of c. In this section we consider that zero-padding might be used within the DFT, meaning that $K \geq N$. More rigorously, this situation translates to saying that we wish to evaluate at C the W-gradient of $\tilde{\mathcal{E}}$, defined as:

$$\begin{aligned} \tilde{\mathcal{E}}\colon \mathbb{C}^N &\to \mathbb{R} \\ z &\mapsto \mathcal{E}(\mathrm{DFT}^{-1}(z)) \end{aligned} \tag{4.34}$$

According to the chain rule of Wirtinger calculus stated in the previous Section we have for $k \in [0, K-1]$:

$$\begin{aligned} \frac{\partial \tilde{\mathcal{E}}}{\partial z_k}(C) = \sum_{n\in[1,N]} &\left(\frac{\partial \mathcal{E}}{\partial z_n}(\mathrm{DFT}^{-1}(C)\right) \times \frac{\partial \mathrm{DFT}_n^{-1}}{\partial z_k}(C) \\ &+ \left(\frac{\partial \mathcal{E}}{\partial \overline{z_n}}(\mathrm{DFT}^{-1}(C))\right) \times \frac{\partial \overline{\mathrm{DFT}_n^{-1}}}{\partial z_k}(C) \end{aligned} \tag{4.35}$$

But DFT^{-1} is $\mathbb{C}$-differentiable, meaning that according the properties of Wirtinger calculus its $\overline{W}$-gradient is null. From (4.31) then follows that the W-gradient of $(\mathrm{DFT}^{-1})^*$ is also null, which leaves us with:

$$\frac{\partial \tilde{\mathcal{E}}}{\partial z_k}(C) = \sum_{n\in[1,N]} \frac{\partial \mathcal{E}}{\partial z_n}(c) \times \frac{\partial \mathrm{DFT}_n^{-1}}{\partial z_k}(C) \tag{4.36}$$

From (2.4) we easily derive:

$$\begin{aligned} \frac{\partial \mathrm{DFT}_n^{-1}}{\partial z_k}(C) &= \frac{1}{K}\frac{\partial}{\partial z_k}\left(\sum_n z_n e^{i\frac{2\pi mn}{K}}\right)\Bigg|_{z=C} \\ &= \frac{1}{K} e^{i\frac{2\pi nk}{K}} \end{aligned} \tag{4.37}$$

Which re-injected in (4.36) gives:

$$\frac{\partial \tilde{\mathcal{E}}}{\partial z_k}(C) = \frac{1}{K} \sum_{n\in[1,N]} \frac{\partial \mathcal{E}}{\partial z_n}(c) e^{i\frac{2\pi nk}{K}} \tag{4.38}$$

$$= \frac{1}{K} \overline{\left[\sum_{n\in[1,N]} \overline{\left(\frac{\partial \mathcal{E}}{\partial z_n}(c)\right)} e^{-i\frac{2\pi nk}{K}}\right]} \tag{4.39}$$

Here we recognize a DFT[8], leading to the expression of the whole W-gradient of $\tilde{\mathcal{E}}$ as:

$$\frac{\partial \tilde{\mathcal{E}}}{\partial z}(C) = \frac{1}{K}\mathrm{DFT}\left(\overline{\left[\frac{\partial \mathcal{E}}{\partial z}(c)\right]}\right) \tag{4.40}$$

But since $\mathcal{E}$ (and $\tilde{\mathcal{E}}$) are real-valued functions, according to (4.32) the previous expression can be formulated in a cleaner way:

$$\frac{\partial \tilde{\mathcal{E}}}{\partial \overline{z}}(C) = \frac{1}{K}\mathrm{DFT}\left(\frac{\partial \mathcal{E}}{\partial \overline{z}}(c)\right) \tag{4.41}$$

In other words, knowing the $\overline{W}$ (or equivalently the W)-gradient of a cost function at a given point c in time domain, it is possible to convert it over to the frequency domain to obtain the gradient at the spectrum C.

Alternatively, the expression (4.41) can also be reversed to give:

$$\frac{\partial \mathcal{E}}{\partial \overline{z}}(c) = K \cdot \mathrm{DFT}^{-1}\left(\frac{\partial \tilde{\mathcal{E}}}{\partial \overline{z}}(C)\right) \tag{4.42}$$

Which this time allows us to transition from the gradient of the objective function at C to the gradient at c.

4.3.6 Application example

By combining the properties of Wirtinger calculus with this computationally efficient (thanks to the use of FFT) way of converting gradients between time and frequency domains, it becomes straightforward to compute the gradient of the objective function of our synthesis algorithm with respect to the base time signal. Because expressing the gradient of the whole function would be of little interest, we only present the imposition of a single statistic: in this case, we arbitrarily choose the skewness (or third moment) of the cochlear envelope of a given band.

This case can be formulated as follows: we call s a real-valued array of length N representing the base sound signal we wish to alter using gradient descent and S its DFT also of length N (we do not use zero-padding in the DFT). S is then windowed in the frequency domain (which includes both the cochlear filtering and the analytic extension) by a function $\mathcal{G}$ defined as:

$$\begin{aligned} \mathcal{G}\colon \mathbb{C}^N &\to \mathbb{C}^N \\ S &\mapsto W.S \end{aligned} \tag{4.43}$$

[8]Note that since $N \leq K$, the sum over n in (4.38) cannot be interpreted as an inverse DFT, resulting in the identification of the conjugate of the DFT in (4.39)

With $W \in \mathbb{C}^N$ the spectral weighting array used for band-filtering S and . the element-wise product. We denote the filtered spectrum $B = \mathcal{G}(S)$, and $b = \mathrm{DFT}^{-1}(B)$ its inverse DFT of length N: b is thus the analytic extension of a cochlear sub-band (selected by W) of s. In this example we want to impose a given value γ to the third order moment of the envelope of b, so the cost function $\mathcal{C}$ is chosen as the squared distance between the third order moment of the envelope of $|b|$ the sub-band and γ:

$$\begin{aligned} \mathcal{C} \colon \mathbb{C}^N &\to \mathbb{R} \\ b &\mapsto \left[\left(\sum_{n \in [0, N-1]} |b_n|^3 \right) - \gamma \right]^2 \end{aligned} \tag{4.44}$$

For simplicity's sake we will consider the raw moment instead of the usual standardized moment that can be found in [McDermott and Simoncelli 2011]. Using the rules of Wirtinger's calculus detailed in Section 4.3.4 we straightforwardly obtain the $\overline{W}$-gradient of $\mathcal{C}$ with respect to the sub-band at b:

$$\frac{\partial \mathcal{C}}{\partial z}(b) = 3 \left[\left(\sum_{n \in [0, N-1]} |b_n|^3 \right) - \gamma \right] b.\,|b| \tag{4.45}$$

But since the gradient descent is to be performed over the base signal s we need to convert the gradient at b over a gradient at s. Mathematically speaking, we wish to know the $\overline{W}$-gradient of the function $\tilde{\mathcal{C}}$ defined as:

$$\begin{aligned} \tilde{\mathcal{C}} \colon \mathbb{C}^N &\to \mathbb{R} \\ s &\mapsto \mathcal{C}\left(\mathrm{DFT}^{-1}(\mathcal{G}(\mathrm{DFT}(s))) \right) \end{aligned} \tag{4.46}$$

Since we know the gradient of $\mathcal{C}$, all we need to do is convert it to the frequency domain, compose it with $\mathcal{G}$ and bring it back to the time domain using the rules of Wirtinger calculus and time-frequency gradient conversion.

Using the time to frequency domain conversion expression in (4.41) we obtain the value of the $\overline{W}$-gradient of $\mathcal{C} \circ \mathrm{DFT}^{-1}$ at B as:

$$\frac{\partial \left(\mathcal{C} \circ \mathrm{DFT}^{-1} \right)}{\partial \overline{z}}(B) = \frac{1}{N} \mathrm{DFT} \left(\frac{\partial \mathcal{C}}{\partial z^*}(b) \right) \tag{4.47}$$

From here we need to obtain the gradient of $\mathcal{C} \circ \mathrm{DFT}^{-1} \circ \mathcal{G}$. Since $\mathcal{G}$ is a simple element-wise product, and since as stated in Section 4.3.4 Wirtinger derivation is linear, we directly obtain:

$$\frac{\partial \left(\mathcal{C} \circ \mathrm{DFT}^{-1} \circ \mathcal{G} \right)}{\partial \overline{z}}(S) = W. \frac{\partial \left(\mathcal{C} \circ \mathrm{DFT}^{-1} \right)}{\partial \overline{z}}(B) \tag{4.48}$$

All that is left now is the conversion from frequency to time domain to obtain the $\overline{W}$-gradient at s. Using the result in (4.42) gives:

$$\frac{\partial \tilde{\mathcal{C}}}{\partial \overline{z}}(s) = N \cdot \mathrm{DFT}^{-1} \left(\frac{\partial \left(\mathcal{C} \circ \mathrm{DFT}^{-1} \circ \mathcal{G} \right)}{\partial \overline{z}}(S) \right) \tag{4.49}$$

Combining (4.47), (4.48) and (4.49) then gives:

$$\frac{\partial \tilde{\mathcal{C}}}{\partial \overline{z}}(s) = \mathrm{DFT}^{-1}\left(W.\mathrm{DFT}\left(\frac{\partial \mathcal{C}}{\partial \overline{z}}(b)\right)\right) \tag{4.50}$$

This relation now explicitly links the $\overline{W}$-gradient at b expressed in (4.45) to the $\overline{W}$-gradient at s, the base time signal. Since s is a purely real-valued signal, we can then make use of the previous result to obtain the more common real gradient of $\tilde{\mathcal{C}}$ at s. Indeed, from the definition of the $\overline{W}$-derivative in (4.21) and since $\tilde{\mathcal{C}}$ is real-valued, we have that if $s \in \mathbb{R}^M$:

$$\frac{\partial \tilde{\mathcal{C}}}{\partial x}(s) = 2\,\mathrm{Re}\left\{\frac{\partial \tilde{\mathcal{C}}}{\partial \overline{z}}(s)\right\} \tag{4.51}$$

Which results in the final expression of the real gradient of the cost function with respect to the real base time signal:

$$\frac{\partial \tilde{\mathcal{C}}}{\partial x}(s) = 6\left[\left(\sum_{n\in[0,N-1]} |b_n|^3\right) - \gamma\right] \mathrm{Re}\left\{\mathrm{DFT}^{-1}\left(W.\mathrm{DFT}\left(b.\,|b|\right)\right)\right\} \tag{4.52}$$

It is important to note that Wirtinger calculus is not strictly speaking necessary to express this gradient. Since in our practical case $\tilde{\mathcal{C}}$ is eventually a function of $\mathbb{R}^N \to \mathbb{R}$, it is possible to express its output as a function of the base time signal and without using complex values. From there we could always compute its gradient with respect to the base time signal. The downside of this is that we often end up with overly complex expressions, which derivatives are a hassle to compute. As an example, the skewness of the cochlear envelope of the sub-band b from which $\tilde{\mathcal{C}}$ is computed can be expressed as:

$$\sum_{n\in[0,N-1]} \sqrt{\frac{1}{N}\left(\sum_{k\in[0,N-1]} A_k C_{n,k} + B_k D_{n,k}\right)^2 + \left(\sum_{k\in[0,N-1]} B_k C_{n,k} - A_k D_{n,k}\right)^2}^3 \tag{4.53}$$

with:

$$\begin{cases} A_k = \mathrm{Re}(W_k) \\ B_k = \mathrm{Im}(W_k) \\ C_{n,k} = \sum_{m\in[0,N-1]} s_m \cos\left(2\pi\frac{k(n+m)}{N}\right) \\ D_{n,k} = \sum_{m\in[0,N-1]} s_m \sin\left(2\pi\frac{k(n+m)}{N}\right) \end{cases} \tag{4.54}$$

And while it is entirely possible (although potentially not very elegant) to compute the derivative of a function involving this expression, computing it straightforwardly means that it might be tedious to put it back under a form such as (4.52).

Setting this aside, we can now explicitly express the gradient of the objective function $\tilde{\mathcal{C}}$ with respect to the base time signal s using the chain rule of Wirtinger calculus, and despite this function being computed from the spectrum S of s. Through this example, we wish to illustrate the fact that the computation of such a gradient is easily doable for any of statistics that are part of our parametrization. Additionally, the objective functions of each statistics can be summed into a global objective function, which gradient is the sum of each objective function gradient: this allows us to not only perform the statistics imposition directly onto the base time signal, but also to impose all statistics at once. With this, we are able to get rid of the convoluted imposition onto the cochlear envelopes proposed in [McDermott and Simoncelli 2011] and replace it with a simpler and more effective statistics time domain imposition. The new imposition is illustrated on Figure 4.3.

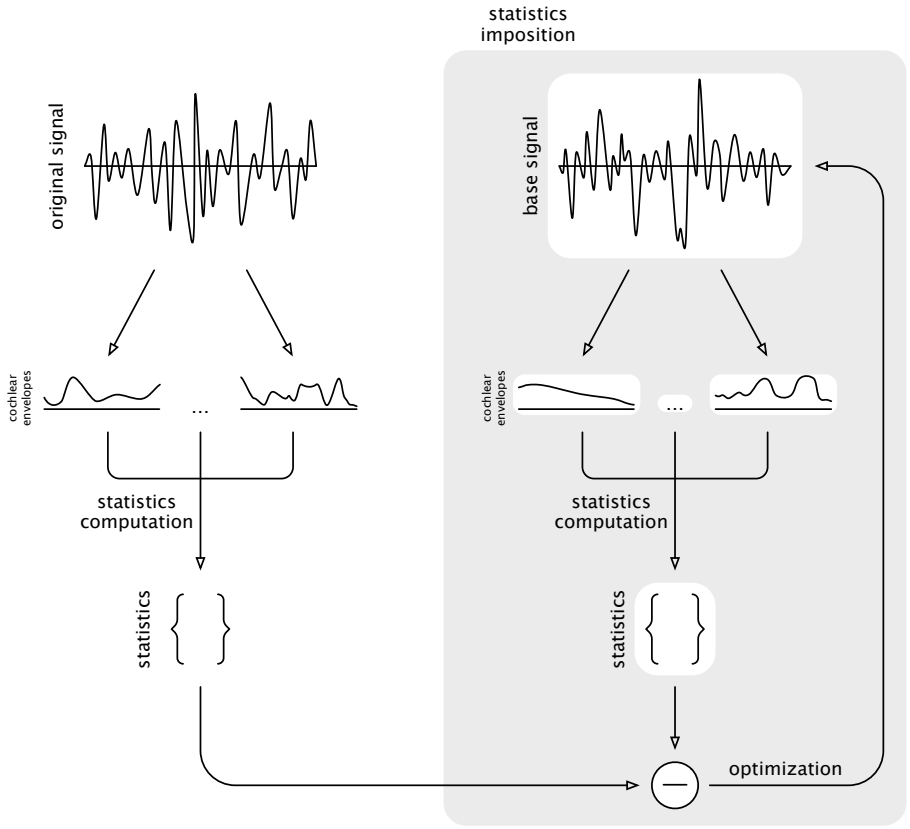


Figure 4.3: *Imposition of the statistics from the original signal onto a base signal. The envelope of all cochlear sub-bands are extracted and their statistics (detailed in Figure 4.1) are computed. The base time signal is then optimized until those statistics are close to those of the original signal. Unlike the imposition described in Figure 4.2, this process imposes all statistics at once.*

This work is presented in [Caracalla et al. 2017], our paper that was accepted at the 20$^{\text{th}}$ International Conference on Digital Audio Effects (DAFx17).

4.3.7 Results

Textures synthesized using this time imposition are available at `http://recherche.ircam.fr/anasyn/caracalla/thesis/mcderm_imp.php`. Despite it simplifying the process of statistics imposition and allowing the statistics of the base signal to be closer to those of the target, the results of this algorithm do not represent an improvement on those of the initial method of McDermott & Simoncelli.

4.4 Sharp event synthesis

The main flaws of the original synthesis algorithm, namely its difficulties at synthesizing sharp events and "un-textural" salient events are still present in our modified algorithm. As visible in Figure 4.4 impacts are still not well reproduced along the frequency axis: instead of a vertical line spanning most of the frequency axis, the synthesis only produces shorter lines spanning only parts of the axis.

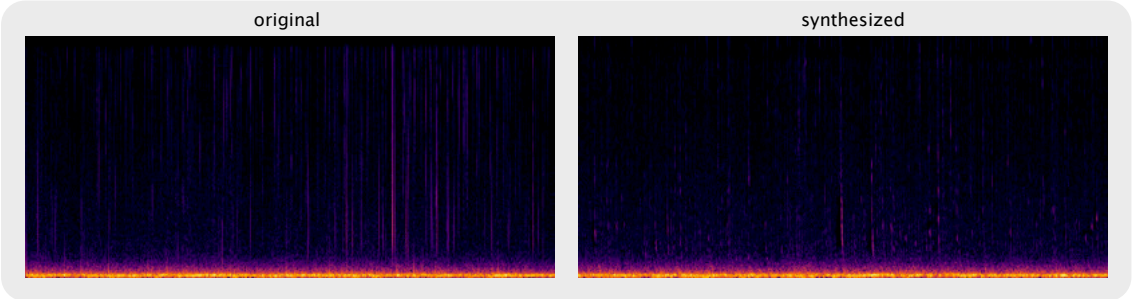


Figure 4.4: *Visual comparison between the log-spectrograms of an original fire texture and the texture synthesized using our time-imposition variant of the perceptual synthesis algorithm from McDermott & Simoncelli.*

We first attempted to tend to the issue of sharp events, and identified two main potential causes at the root of this issue. The first being that the statistics used were not adapted to describing such events, and the second being that the sharpness of the signal was somehow blurred before those statistics were computed. We decided on exploring the second possibility first, and started with investigating the filter-banks used in this synthesis method.

4.4.1 Filter-banks modification

In [McDermott and Simoncelli 2011], both the filter-bank used to compute the cochlear sub-bands and the one used to compute the modulation sub-bands are composed of zero-phase band-pass filters. As their name suggests zero-phase filters are filters which phase is null, thus having a purely real frequency response. This means that once the magnitude spectrum of each filter has been chosen, they are multiplied in the frequency domain to the input of the filter-bank without altering its phase. Since the frequency responses of the filters are real-valued and because of the time/frequency duality, this implies that the impulse responses of

the filters are symmetric in the time domain and thus that those filters are not causal[9].

Since the aim of this algorithm is to emulate the way our brain processes sound textures, it seems odd to use non-causal filters. In addition to this, because their impulse responses are symmetric and because of the set of statistics chosen, the parametrization is identical for a signal and its reverse copy. This means that no difference is made between an event that has a sharp attack and slow decay and one that has a slow attack and sharp decay: seeing as those two events are fundamentally different, this can be seen as a major flaw of the algorithm.

For those reasons, we investigated the use of minimum-phase filters. As explained in [Smith 2007a], those filters are causal and their impulse responses have the fastest decay among all impulse responses out of all causal filters having the same magnitude spectra. In this sense, fastest decay can be expressed as:

$$\sum_{n\in[0,K]} \left|h_n^{min}\right|^2 \geq \sum_{n\in[0,K]} |h_n|^2 \quad \text{with} \quad K \in \mathbb{Z}^+ \tag{4.55}$$

with h^{min} the impulse response of the minimum-phase filter, and h that of any causal filter having the same magnitude spectrum[10]. This can be interpreted as minimum-phase filters having the least amount of delay among causal filters, and thus reacting to impulses as quick as possible: out of causal filters, minimum-phase filters are the least harmful to sharp onsets.

Additionally, and as argued in [Smith 2007a], minimum-phase filters may also be more adapted to audio processing than zero-phase filters. Due the impulse of a zero-phase filters being symmetric, any impulse processed through such a filter will possess a sort of "pre-ring" that will precede the transformed impulse. This pre-ring is audible, and may result in chirping artefacts which resembled those we perceive in synthesized textures containing sharp events. Since the impulse responses of minimum-phase filters rise sharply and decay progressively, such an artefact should not be present in processed sounds.

Those were the reason behind our choice to convert the zero-phase filters of the filter-banks into minimum-phase filters. This conversion is made using the Hilbert transform $\mathcal{H}$, which is the imaginary part of the analytic signal computed using (4.1). Given a magnitude spectrum G, the frequency response H of the corresponding minimum-phase filter is computed as:

$$H = G.\exp(i\Theta) \quad \text{with} \quad \Theta = \mathcal{H}(\ln(G)) \tag{4.56}$$

[9] In this sense, a causal filter is a filter only using the actual and previous values of its input. A non-causal filter also uses the incoming values of the input: because of this, such a filter can obviously not be used in a real-time context.

[10] It may be noted that because h^{min} and h have the same magnitude spectrum, those two sums are equal for $K = \infty$.

Sounds synthesized using this filter-bank are available at http://recherche.ircam.fr/anasyn/caracalla/thesis/mcderm_min.php. Despite the switch to minimum-phase filter-banks, the results are sensibly similar to those obtained using zero-phase filters: for instance, the synthesis algorithm still creates chirping artefacts when synthesizing sharp events. It follows that the use of minimum-phase filters does not noticeably improve the synthesis algorithm. This means that its inability to produce sharp onsets is probably a deeper flaw and may originate from an unfit choice with regard to the set of statistics used as parameters to the synthesis.

4.4.2 Base signal initialization

So as to strengthen this hypothesis and in order to check the consistency of the statistics chosen in [McDermott and Simoncelli 2011], we decided to modify the initial base signal prior to the imposition so that it contained temporal spikes. In the original paper, the base signal is chosen as a white noise: this is useful in that the spectrum of a white noise is evenly spread along both time and frequency axes and thus serves as a neutral initialization to the synthesis. By adding spikes to this noise and synthesizing a texture containing sharp events, our aim was to ease the creation of such events in the synthesized texture. If the texture synthesis was enhanced by the presence of spikes then one could suppose that the chosen statistics were sufficient for sharp onset synthesis, and that the issue may originate from the optimization process in itself. If it did not however, then this could be interpreted as the statistics not being adapted to the description of sharp events and thus harming existing onsets during the imposition process. In our implementation, the spikes are created by simply setting the values of a few individual samples to 3 times the amplitude of the white noise.

The comparison between an original fire texture, and two textures synthesized using our time imposition synthesis is available at http://recherche.ircam.fr/anasyn/caracalla/thesis/mcderm_pk.php. The first of our textures has been synthesized using white noise as initialization for the base signal, the other using spiked white noise. The results of this comparison are disappointing: although the sharp events present in the initial base signal are used to create sharp onsets in the synthesized sound, chirping artefacts are still present. This means that vertical lines in the synthesized texture are still damaged and fragmented compared to the original despite being helped by the presence of such lines in the initialization of the base signal. This in turn lends weight to the idea that the issue with sharp event synthesis lies deeper than the optimization process: this could either be due to an incomplete (or wrong) set of statistics, or to the core idea of using auditory sub-bands statistics for sound texture synthesis itself.

4.5 Partial conclusion

At the time of their finding, those conclusions marked an important shift in our work. From that point on, our two perspective were to either find a better set of parameters or to fundamentally alter the paradigm of this synthesis method.

Finding another suitable set of parameters, or even improving the current one, would have represented a consequent time investment. Because sound textures are not a common subject in psycho-acoustics, it was difficult for us to find leads on what statistics could prove interesting to work with. This meant that our main hope was to empirically explore the statistics space and judge by ear the effect of each choice of parameter set. As mentioned in the coming chapters, this method has its own risk: being the only judges of the quality of a synthesis comes with an inherent bias toward some audio characteristics, which in turns might lead to poor decisions during the tuning of the algorithm. The remedy to that is to perform listening tests on a wide array of listener, but this option is costly in time (and potentially in funds) and as such cannot be employed for each potential set of statistics. For those reasons, finding another or completing the current set of statistics seemed like a lackluster possibility.

This left us with the possibility of fundamentally altering the paradigm behind this synthesis method, for instance by changing the representation of sound from which the statistics are computed (in this case, the cochlear and modulation bands). Such work is for instance undertaken in [Kim et al. n.d.] and yields satisfying results. Instead of following a similar path, we decided to rather try and adapt another parametric method intended for visual textures synthesis which presented impressive results: this method and our adaptation of it are the subject of the following chapter.

References for chapter 4

Bouboulis, P. (2010). "Wirtinger's calculus in general Hilbert spaces". In: *arXiv preprint arXiv:1005.5170* (cit. on pp. 56, 57).

Brandwood, D. (1983). "A complex gradient operator and its application in adaptive array theory". In: *IEE Proceedings F-Communications, Radar and Signal Processing* (cit. on p. 57).

Caracalla, H. and Roebel, A. (2017). "Gradient conversion between time and frequency domains using wirtinger calculus". In: *Digital Audio Effects (DAFx)* (cit. on pp. 62, 82, 122).

Fischer, R. F. (2005). *Precoding and signal shaping for digital transmission*. John Wiley & Sons (cit. on p. 57).

Gatys, L., Ecker, A. S., and Bethge, M. (2015). "Texture synthesis using convolutional neural networks". In: *Advances in neural information processing systems* (cit. on pp. 42, 48, 70, 72–74, 76, 79, 82, 83, 90, 92, 101, 123, 125).

Heeger, D. J. and Bergen, J. R. (1995). "Pyramid-based texture analysis/synthesis". In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (cit. on p. 50).

Kim, H.-S. and Smith, J. (n.d.). "Synthesis of sound textures with tonal components using summary statistics and and all-pole residual modeling". In: (cit. on pp. 42, 66).

Liao, W.-H. (2015). "Modelling and transformation of sound textures and environmental sounds". PhD thesis (cit. on pp. 42, 48).

McDermott, J. H., Schemitsch, M., and Simoncelli, E. P. (2013). "Summary statistics in auditory perception". In: *Nature neuroscience* (cit. on pp. 42, 48).

McDermott, J. H. and Simoncelli, E. P. (2011). "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis". In: *Neuron* (cit. on pp. 42, 48, 49, 52, 54, 55, 60, 62, 63, 65, 71, 82, 85, 88, 107, 108, 112, 115, 122, 123).

McWalter, R. I. and Dau, T. (2013). "Analysis of the auditory system via Sound Texture Synthesis". In: *AIA-DAGA Conference on Acoustics* (cit. on p. 48).

Portilla, J. and Simoncelli, E. P. (2000). "A parametric texture model based on joint statistics of complex wavelet coefficients". In: *International journal of computer vision* (cit. on pp. 42, 50, 70, 71, 73, 123).

Smith, J. O. (2007a). *Introduction to digital filters: with audio applications*. Julius Smith (cit. on p. 64).

Wirtinger, W. (1927). "Zur formalen theorie der funktionen von mehr komplexen veränderlichen". In: *Mathematische Annalen* (cit. on p. 56).

Chapter 5

Synthesis based on CNN statistics

Chapter overview

After focusing on perceptually-based parametrization, a CNN-based parametrization for visual synthesis is investigated. In this paradigm, the parameters are the cross-correlations between the feature maps of each convolutional layer of a trained CNN. After interpreting its working, we adapt this method to work on log-spectrograms in order to synthesize sound textures. The synthesis is performed directly in the time domain, using an untrained CNN. Results are showcased at the end of the chapter.

Contents

After reaching the conclusions on synthesis via perceptual statistics presented in the previous chapter, we decided to focus on a method yielding impressive results in visual texture synthesis. This method, presented in [L. Gatys et al. 2015], is a successor to the parametric visual texture synthesis introduced in [Portilla et al. 2000] and leads to the successful style transfer method presented in [L. A. Gatys et al. 2016].

So as to introduce our adaptation of this algorithm to audio texture synthesis, the following section details its working in greater lengths than the explanation given in Chapter 3.

5.1 Explanation of the visual algorithm

The synthesis method presented in [L. Gatys et al. 2015] belongs to the category of parametric syntheses. As such, it works in a similar fashion (besides the shift from 1D to 2D) as the algorithm presented in the previous chapter: a set of summary statistics is extracted from an existing texture, and a base[11] image is modified until it fits those statistics. Its main difference lies in the fact that instead of being directly computed from a perceptually-based representation, those statistics are derived from the feature maps of a CNN[12].

5.1.1 Analysis

The representation used in this synthesis method is the simple 3D matrix of the images storing the values of each pixel, with the first two dimensions being the two spatial axes of the image and the third being the color depth (usually this depth is either of 1 for black and white images, or 3 for colored images). Instead of being put on the representation, the burden of the parametrization is entirely transferred to the statistics extraction performed using a CNN.

CNN architecture

The CNN used in [L. Gatys et al. 2015] is the VGG-19 network, a network trained in image recognition and introduced in [Simonyan et al. 2014]. The convolutional part of its architecture can be broken down into blocks of similar organization. The first part of each block consists in a succession of convolutional layers with filters of size 3×3 (and as deep as the input of the layer), strides of $(1, 1)$ and padding so that the feature maps have the same size as the input of the layer, each followed by a ReLU activation function. The second part of each block consists in a 2×2 max-pooling layer with stride $(2, 2)$, which effectively reduces the size of the input by 2. There are a total of 5 blocks, holding from 2 to 4 convolutional layers each, while the number of filters each layer possesses goes from 64 to 512 for the deepest layers.

[11] Like in the previous chapter, the term "base" refers to the signal that is being modified throughout the synthesis process.

[12] See Section 2.3 for a quick introduction to CNNs.

Statistics extraction

Similarly to [Portilla et al. 2000] and [McDermott and Simoncelli 2011], the statistics used as parameters are correlations. Instead of using the correlations between the outputs of a linear filter-bank, the authors use the correlations between the feature maps of the CNN, which can be interpreted as the outputs of a non-linear (due to the action of the activation layers) succession of filter-banks. Given an input image to the network, those correlations are stored in Gram matrices[13], one for each convolutional layer. The Gram matrix G^l of the l^{th} layer is of size $N_l \times N_l$, with N_l the number of filters in the layer, and its elements are computed as follows:

$$G^l_{ij} = \sum_{m,n} F^l_{imn} F^l_{jmn} \quad \text{with} \quad (i,j) \in [0, N_l - 1]^2 \tag{5.1}$$

with F^l_{imn} the element at position (m,n) of the i^{th} feature map of the l^{th} layer.

As such, the matrix G^l is simply a convenient way of storing the cross-correlations at shift $(0,0)$ between all the feature maps of the l^{th} layer (also including the auto-correlations of those feature maps). This means that for a CNN containing L convolutional layer, the correlations used as parametrization are all contained within an array of Gram matrices $(G^0, ..., G^{L-1})$.

As a side note, the fully connected layers of the CNN are not used in any way: this parametrization focuses on the feature extraction properties of the convolutional layers, and not on the classification abilities of the dense layers.

5.1.2 Synthesis

Unlike the original synthesis algorithm proposed in [McDermott and Simoncelli 2011], the (re-) synthesis of a visual texture is done by imposing all parameters at once. This is done by defining a global objective function containing all statistics that are to be minimized.

Texture Loss

To do so, and given a target texture, this objective function (henceforth called texture loss) is set as a weighted distance between the gram matrices of the target and those of the base image, over which the imposition is to be performed. This loss is defined as:

$$\mathcal{L} = \sum_{l=0}^{L-1} \frac{1}{N_l^2 M_l^2} \sum_{i,j} \left(\hat{G}^l_{ij} - G^l_{ij} \right)^2 \tag{5.2}$$

with M_l the number of elements in each feature map of the l^{th} layer, and G^l the l^{th} gram matrix with the base image as input to the network while $\hat{G}^l$ is the l^{th} gram matrix with the target texture as input.

It is important to keep in mind that this loss is meant as a parametric loss between the base image and the original texture: at no point will it serve to train the CNN, or modify it in any way.

[13] Generally speaking, the Gram matrix G of a set of vector $v_1, .., v_N$ is the matrix which element G_{ij} is the product of v_i and v_j.

Statistics imposition

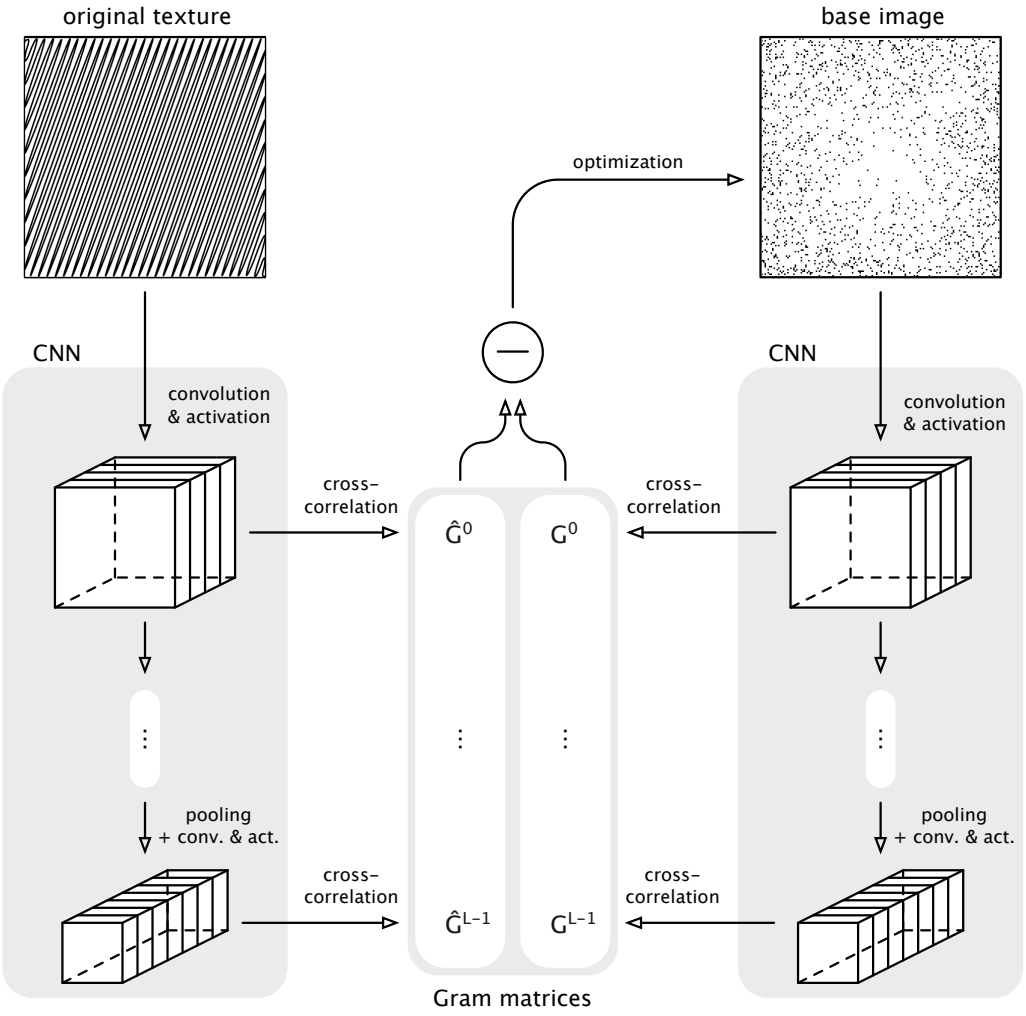


Figure 5.1: *Imposition of the statistics from the original visual texture onto a base image. Both images are passed through the same CNN, and for each of them the cross-correlations between the feature maps of each layer are computed. Those cross-correlations are stored inside sets of Gram matrices. The base image is then iteratively modified until its set of Gram matrices resemble that of the original texture.*

The synthesis process is performed over the base image, which is usually initialized as a white noise image. Since the texture loss $\mathcal{L}$ is computed from the base image using convolutions and pseudo-differentiable functions (i.e. the activation and pooling functions), its gradient with respect to the base signal is easily computed. In addition to this, the computational efficiency of common deep learning libraries (such as tensorflow) allow for fast and automatic gradient computations. Regarding the optimization of the texture loss, the method employed in [L. Gatys et al. 2015] is the L-BFGS-B mentioned in Section 2.3.2.

By optimizing the texture loss $\mathcal{L}$, the correlations between the feature maps computed from the base image are modified all at once to resemble those of the original texture. The process is illustrated in Figure 5.1.

5.1.3 Results

Examples of visual textures synthesized using this method are shown in Figure 5.2. Although an in-depth presentations of those results does not enter the scope of this, it is notable that at the time of the publication of [L. Gatys et al. 2015], those results represented a consequent improvement over the state of the art.

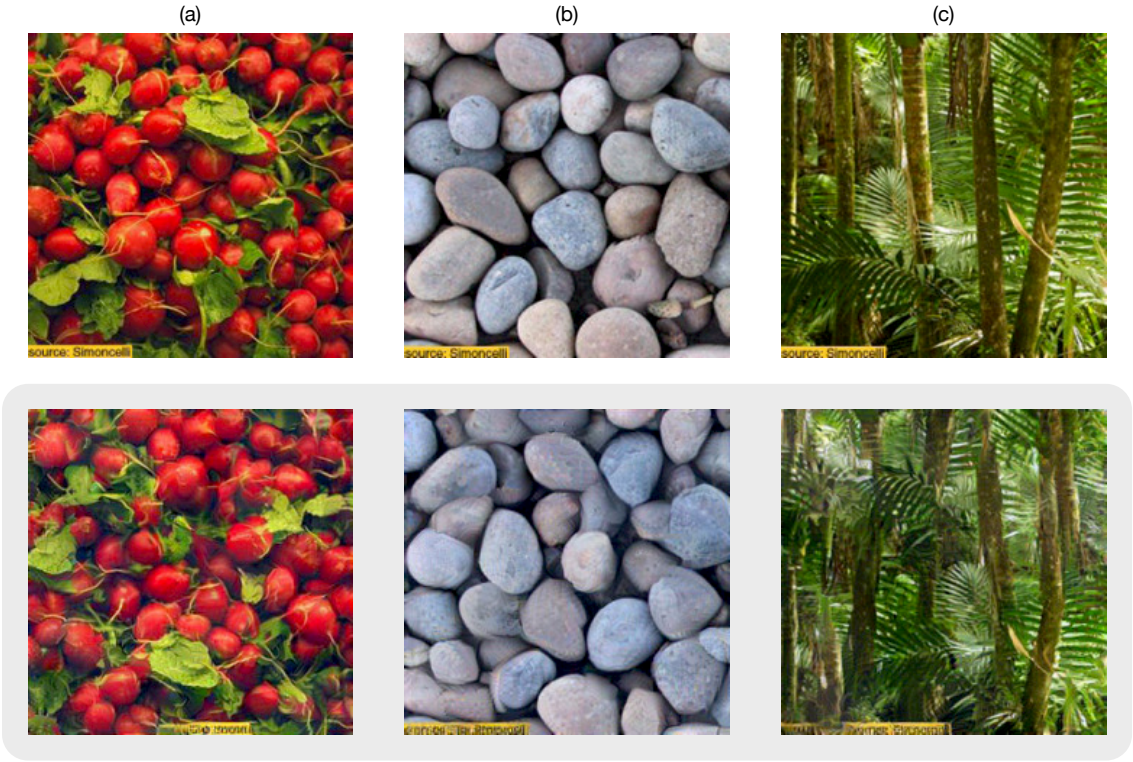


Figure 5.2: *Examples of visual textures: (a) arabesque pattern, (b) wild moss, (c) zebra fur, (d) wood grain, (e) dried earth and (f) water surface.*

5.2 Interpretation

The original article does not provide extensive insight on the reasons for the success of its synthesis methods, save for introducing it as a successor to the method presented in [Portilla et al. 2000] and exchanging the linear filter-bank for a non-linear one. For this reason, this section presents our attempts at understanding the mechanisms at work behind it in order to then adapt it to sound texture synthesis.

5.2.1 On the role of filters

Filters are the base elements of convolutional layers: their convolution with the input of the layer, followed by the non-linearity of the activation layer, highlights the portions of the input that matches the content of the filter. But our paradigm of interest here is slightly different from the usual "input $\rightarrow$ output" one, and is rather a backward take on it: the input is modified until a given output is met. In this context, what influence does a filter have ?

To answer this, and taking inspiration from [Erhan et al. 2009] it is easy to visualize what a filter "wants" to see in an input. To do so, we define for a given filter i the following function:

$$\mathcal{I}_i = \sum_{m,n} F_{imn}^2 \tag{5.3}$$

with F_k the k^{th} element of the flattened feature map of said filter. $\mathcal{I}_i$ is then simply the energy of the feature map of the i^{th} filter, or its auto-correlation function at shift $(0, 0)$. Given an initialization image (for instance a white noise image), it is possible to perform an optimization on it so as to maximize $\mathcal{I}_i$. The result of this optimization is an image that strongly activates the filter, and thus represents the pattern that the filter is tuned to detect in an way that is easy to understand (since this result is still in the same space as the inputs of the network).

In practice, we add a constraint to the optimization stating that the values of the image needs to be contained in an arbitrary interval (e.g. $[-1, 1]$): this is done to prevent the optimization from simply increasing the values of the image *ad lib* in order to increase the values of the feature maps. For lack of an existing name, we call those images the identikits[14] of their respective filters.

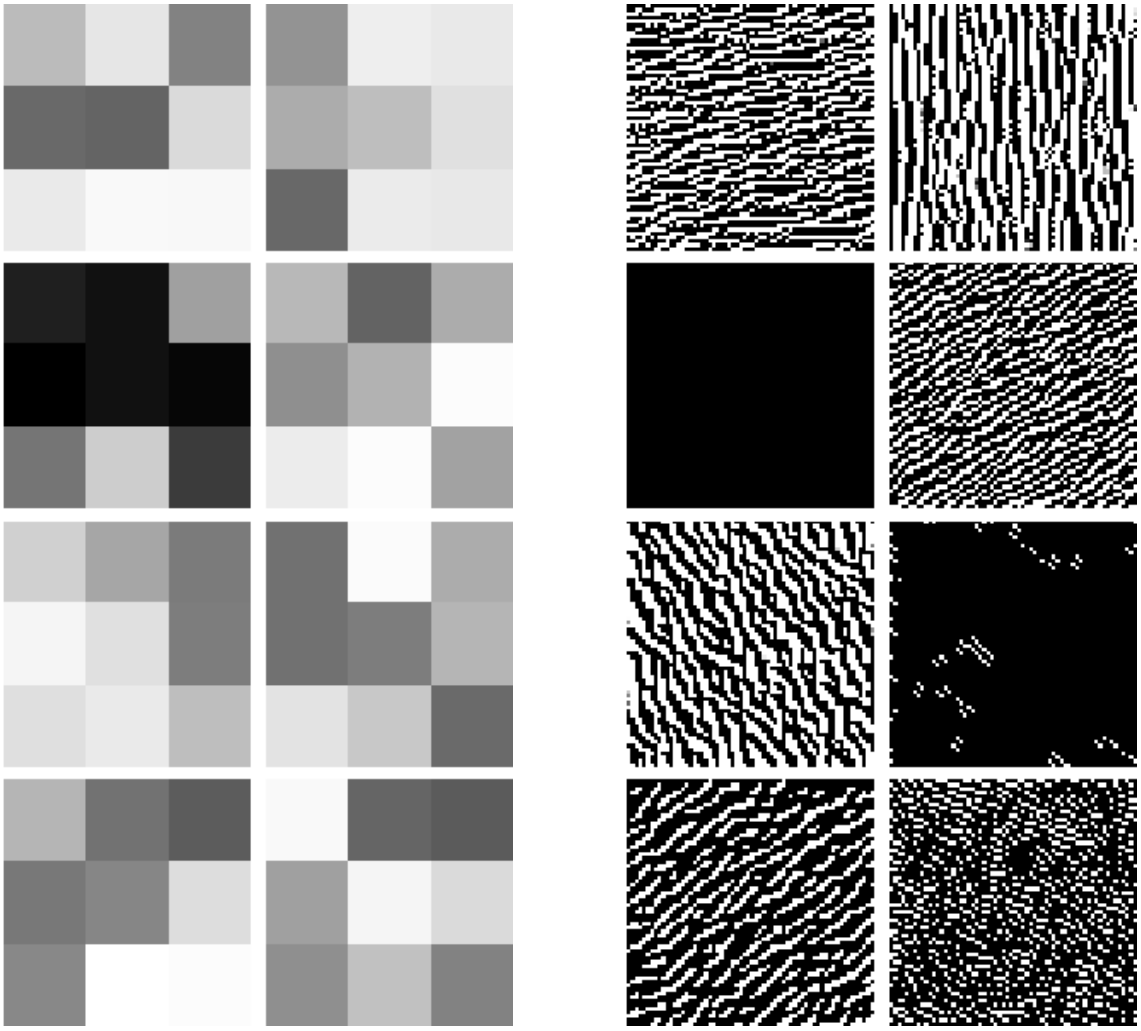
As an example, the identikits of 8 filters of size 3×3 are shown on Figure 5.3. Those filters are taken from the first layer of a simple multi-layered CNN, trained on recognizing various sounds (e.g "music", "traffic", etc.) from their spectrograms: this task can be seen as the audio equivalent of the image recognition task of the CNN used in [L. Gatys et al. 2015]. The pattern that each filter looks for can be deduced from the values of said filters: if the filter contains a row of high values its identikit will contain vertical stripes, while if it contains similar values its identikit will be a constant matrix.

Since the diagonal of a Gram matrix as defined in Eq.(5.1) contains the functions $\mathcal{I}_i$ (as defined in Eq.5.3) of every filter i of its layer, identikits bring a first element of response to the process at work behind the CNN-based visual texture synthesis algorithm : the Gram matrices of the target texture contain in their diagonals a description of the patterns present in the texture, with each value indicating the amount of presence of a given pattern. Since those values are averages over the whole feature maps, they do not however describe the location of those patterns. This is a desired effect, since the goal of texture synthesis is not to reproduce a copy of the target but a texture possessing the same properties: as such, the algorithm should be able to synthesize elements resembling those of the target but at different positions.

5.2.2 On the role of depth

Although describing a texture by how present some patterns are in it is possible, the size of those patterns is a critical issue. If we for instance imagine a texture containing mostly circles, describing it with patterns smaller than those circles (e.g. patterns of circle segments) would lead to a description no different from that of a texture made of wavy lines: it is thus necessary for the patterns used for the description to

[14] A synonym of composite portrait, and a reference to an infamous music band.



(a) (b)

Figure 5.3: *In (a): filters of size* 3×3 *from the first layer of a CNN trained on spectrograms for acoustic scene detection. In (b): their respective identikits of size* 64×64*, showing the pattern each filter is looking for in an input image.*

be large enough to contain those present in the texture.

As noticeable on Figure 5.3, the characteristic size of the identikits is of the same order as the size of the filters: this is simply due to the fact that the convolution of those filters with the input is a local operation, and does not take into account any value outside of the 3×3 area that is being multiplied with the filter. For this reason and in order to properly describe a texture, one would need to use filters of size equivalent to the biggest patterns present in it.

This, however, can be circumvented by using filters with large effective receptive field (ERF). The ERF, as detailed in [Luo et al. 2016], is the size of the zone of the input matrix that a filter "sees". In the first layer of a CNN, this size is equivalent to the size of the filter. This changes when venturing in deeper layers of a CNN. Each value of the feature map outputted by the first layer is computed using several values of the input, and thus represents them in an indirect way. The following layer is thus

fed values that correspond to an area of the input of the network that is larger than the simple size of its filters. This idea is illustrated in Figure 5.4.

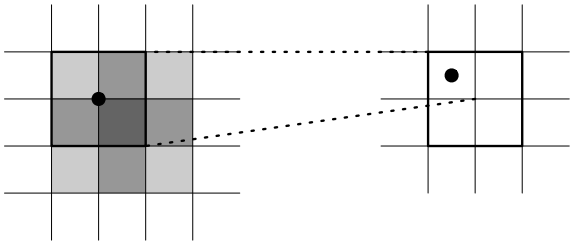


Figure 5.4: *Illustration of the ERF (in gray) of a* 2×2 *filter succeeding to a* 2×2 *filter from the previous layer: because of their sizes, the area of the feature map (on the right) that is convolved with the second filter is equivalent to a wider area of the input (on the left) than the* 2×2 *size of the second filter.*

The deeper the convolutional layer, the more the ERF increases in size[15]. This increase is even more drastic when down-sampling layers are used. Because it shrinks the feature map that it takes as input, a down-sampling layer multiplies by a given factor the size of the ERF of the next layer.

Going back to the matter at hand, this implies that it is possible to describe large patterns by using deep convolutional layers instead of large filters. This is illustrated on Figure 5.5 where the identikits of filters of different layers are showcased: all belong to the same network used for Figure 5.3 which architecture is an alternation of 3×3 convolution layers and 2×2 average-pooling layers. Despite all filters being of the same size, the patterns they describe are larger and more complex the deeper the layer is. One can also note that identikits of deep layers filters are more reminiscent of the visual properties of spectrograms, which is to be expected since the features those filters are trained to detect are of a higher level (both semantically and in size) than those of the first layer: the odds of them being tuned to detect a feature that is meaningful to us are thus also higher.

This interpretation is corroborated by the mention in [L. Gatys et al. 2015] that using only the first layers of their deep CNN results in the synthesis of images presenting the right low-level characteristics (such as color) but no proper organization on a higher level. The patterns present in the synthesis get more and more faithful to the original when including the Gram matrices of deeper layer in the parametrization.

5.2.3 On the role of cross-correlations

Our reasoning until this point supposes that the CNN used possesses enough pertinently specialized filters that the patterns they represent are enough to describe any texture we may want to synthesize. It is unlikely however that this whole array of texture is present in the data-set used for the training phase of the CNN, and as

[15] In practice, not all input values that are part of the ERF are taken into account with the same weight (as denoted by deeper grays in Figure 5.4): this results in ERF being in practice smaller than one might think, as explained in [Luo et al. 2016].

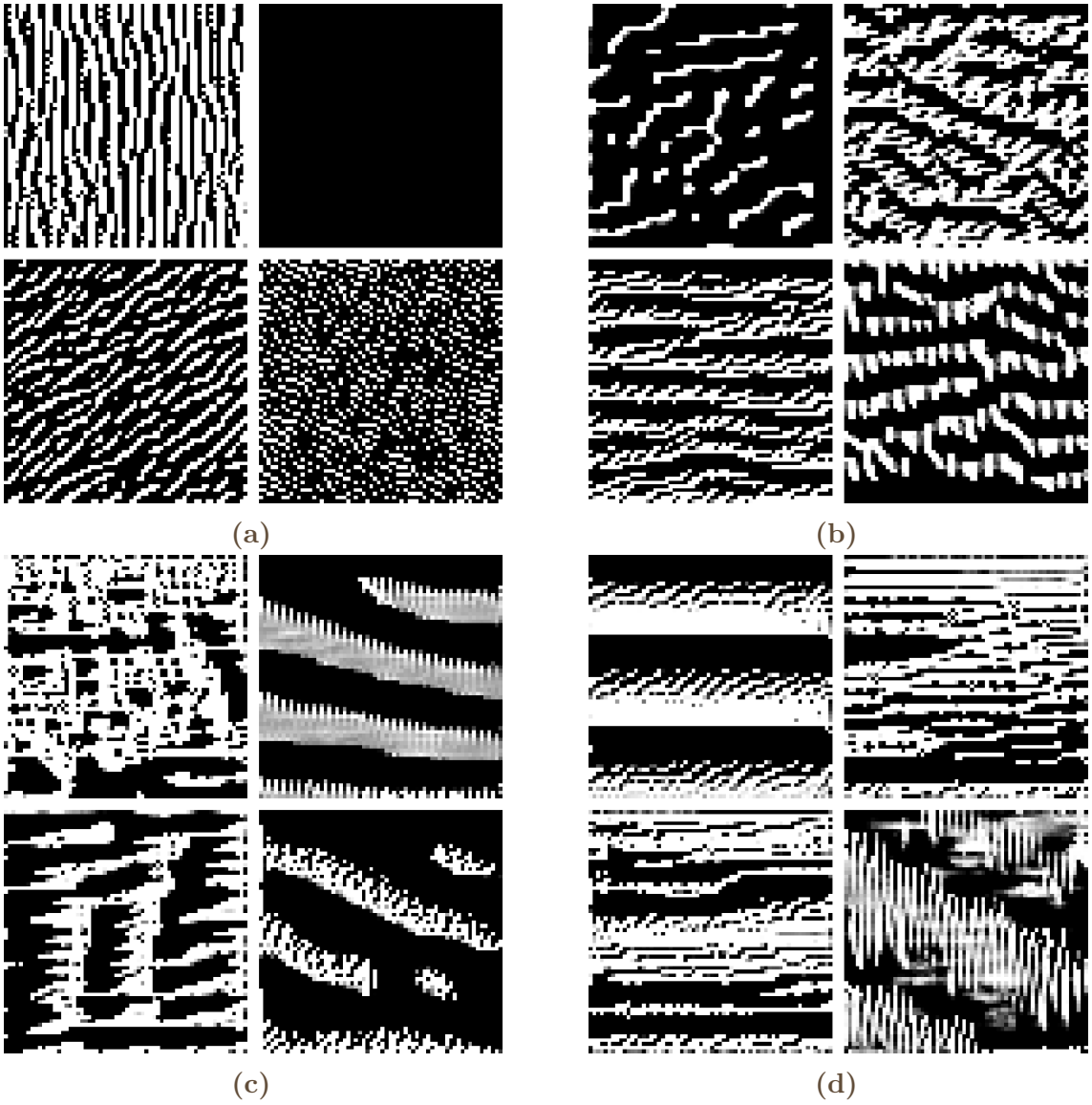


Figure 5.5: *Identikits of some* 3×3 *filters of a deep CNN trained on spectrograms for sound recognition. In (a): layer 1. In (b): layer 2. In (c): layer 3. In (d): layer 4.*

such that its filters are specialized in all patterns we might want to reproduce.

The solution to this is to not only use the auto-correlations at shift $(0, 0)$ of the feature maps, stored in the diagonals of the Gram matrices, but also their cross-correlations at shift $(0, 0)$, stored in the rest of those matrices (and expressed as in Eq.(5.1) with $i \neq j$). This cross-correlation value being high means that the activations of the two filters are mostly simultaneous, and we might suppose that this is indicative of the presence of a pattern "in between" the two that those filters are tuned to.

This intuition is confirmed by visualizing the image that maximize a given cross-correlation between the feature maps of a pair of filters along with the identikits of those filters, as shown Figure 5.6. Those images are obtained in a similar fashion to the identikits showcased in Figures 5.3 5.5, and can be considered as the identikits

of a correlation of filters: a white noise image is optimized so as to maximize the cross-correlation of a pair of feature maps while still being arbitrarily bounded to the $[-1, 1]$ interval.

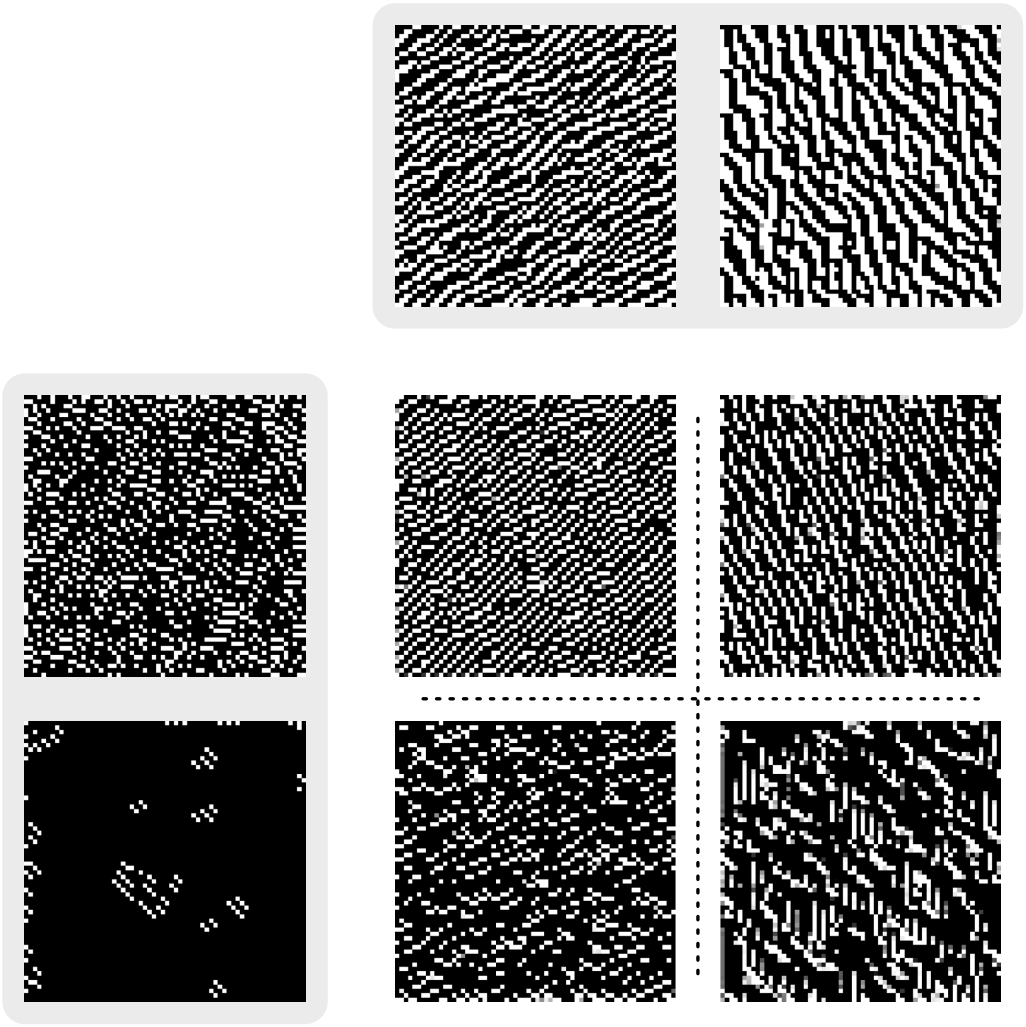


Figure 5.6: *Identikits of the correlations of a few 3×3 filters taken from a deep CNN trained on spectrograms for sound recognition. The identikits of the individual filters are presented at the top and left in grey boxes, while those of the correlations are presented at the intersections.*

As visible in the resulting images, the patterns present in the identikits of the correlations may indeed be interpreted as being "in between" the identikits of each filter: for instance, the identikit of the correlation of one filter which identikit is dotted and another which identikit is striped presents finer stripes. As such, the description contained in the Gram matrices of a CNN is not only that of the presence of patterns found in the identikits of its filters, but also of the patterns in between them. This makes the description all the more complete, and allows for the synthesis of patterns that the filters may not be individually specialized in.

5.2.4 On the necessity of using a trained deep CNN

The depth and the training of the CNN used for visual texture synthesis are presented in [L. Gatys et al. 2015] as crucial for the success of the synthesis, since both make for a powerful and meaningful feature space. However, it would appear from the previous sections that the main necessity in the case of texture synthesis is the ability to describe a wide array of patterns of varying sizes: as such, it may be possible to compensate for the training of the CNN (and thus the pertinence of its filters) by simply using a greater number of random filters, and to compensate for the depth of the network (and thus the big ERF of its filters) by increasing their sizes.

This idea is demonstrated in [Ustyuzhaninov et al. 2016], which produces results of the same perceptual quality as those obtained by [L. Gatys et al. 2015] despite using mono-layer untrained CNNs. As we may expect, the network producing the best results includes a wide array of filters of varying sizes ranging from $(3, 3)$ to $(55, 55)$ drawn from a uniform distribution.

As such, texture synthesis using parameters extracted by a random CNN may not qualify as being properly part of the machine learning field: in essence, this parametrization may be seen as the projection of the input texture onto an array of patterns with all positional information being blurred.

5.3 Adaptation to sound textures

The interpretation given in the previous sections may be seen as an alternative to the more classic non-linear filter-bank interpretation, and is substantiated by the work presented in [Ustyuzhaninov et al. 2016]. It also motivated our adaptation of the visual texture synthesis algorithm introduced in [L. Gatys et al. 2015] into a sound texture synthesis algorithm. The following section details our reasoning during the building of our initial sound texture synthesis algorithm based on CNN statistics, as detailed (albeit slightly more shallowly) in our article published at the 22$^{\text{nd}}$ International Conference on Digital Audio Effects (DAFx19), [Caracalla et al. 2019]: taking the CNN-based visual synthesis algorithm as base, we go over the tuning needed for it to be used for sound texture synthesis.

5.3.1 Representation

Using a method similar to that of [L. Gatys et al. 2015] requires using a representation in which repeated patterns may be observed throughout the texture. As mentioned in Section 2.2, time-frequency representations of sound textures may broadly be considered as visual textures themselves, despite presenting notable differences in behavior. For this reason, we opt for using spectrograms as time-frequency representation. Most of the sounds worked with during this thesis are sampled at 22050 Hz, and we arbitrarily decide on a window length of 512 samples (around 23 ms) and a hop-size of 256 samples.

Because the original synthesis algorithms operates on patterns that are noticeable to the human eye, our naive approach is to compress the spectrograms so that their patterns are more easily observable. To do so, we first divide every spectrogram by its maximum (so that its new maximum is now 1), and compress it using the following:

$$S = \frac{\log(1 + CX)}{\log(1 + C)} \tag{5.4}$$

with X the original spectrogram, S the compressed spectrogram (or log-spectrogram) and C the compression factor. The higher C is, the more the high values of the spectrogram are compressed. Based on our observations, we decide on a value of 1000 for C. This compression also guarantees that the values of the compressed spectrogram spans the range $[0, 1]$. The visual comparison between uncompressed and compressed spectrogram is shown on Figure 5.7.

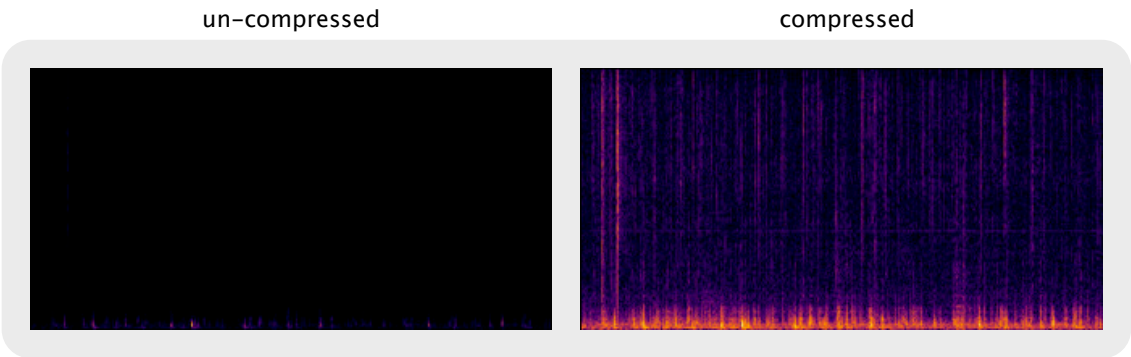


Figure 5.7: *Side-by-side comparison of the un-compressed spectrogram and compressed spectrogram (or log-spectrogram) of a fire recording.*

5.3.2 CNN architecture

Following the idea developed in [Ustyuzhaninov et al. 2016], we opt for the use of a single-layer random CNN. This also presents the obvious advantage of not needing to train a network. The architecture parameters were chosen arbitrarily as a first attempt at sound texture synthesis, and are discussed more in-depth in the following chapter.

The single layer of the CNN is made of 128 square filters of each of the sizes $[3, 5, 7, 11, 15, 19, 23, 27]$ with a stride of $(1, 1)$ and zero-padding. Th zero-padding is chosen so that the results of the differently-sized convolutions can then be stacked. This convolution is followed by the rectified linear unit (ReLU) activation function. The weights of the filters are drawn from a uniform distribution between -0.05 and 0.05, and no bias is applied.

For generalization's sake, the rest of this section is nonetheless presented with a network having K layers (with K being potentially more than 1), although it stays valid when using a single-layer network.

5.3.3 Parametrization

As evoked in Section 2.2, one of the main differences between visual textures and the time-frequency representation of sound textures lies in their different invariances. While visual textures are invariant in both dimensions, seeing as they both represent a spatial dimension, sound textures are only invariant with regard to the time dimension. For this reason, using the Gram matrices defined in Eq. (5.1) to parametrize sound textures is nonsensical: were we to synthesize sound textures using Gram matrices, patterns present in the original texture would be synthesized with unpredictable frequency shifts, destroying the fidelity of the synthesis.

For this reason, we propose the following parameter matrices to replace the Gram matrices:

$$H^l_{ijm} = \sum_n F^l_{imn} F^l_{jmn} \tag{5.5}$$

with F^l_{imn} the feature map at position (m, n) of the i^{th} filter of the l^{th} layer. The resulting 3D matrix[16] discards any time information contained in the feature maps because of the sum over the time dimension, while discriminating between frequency bins via its third dimension.

Given this parametrization, we then define our sound texture loss that acts as a measure of how a base sound resemble a target texture:

$$\mathcal{L} = \sum_l \frac{\|\hat{H}^l - H^l\|_2}{\|\hat{H}^l\|_2} \tag{5.6}$$

with H^l the l^{th} parameter matrix with the base sound as input to the network while $\hat{H}^l$ is the l^{th} parameter matrix with the original texture as input. Minimizing this loss is thus equivalent to imposing the parameters of the original texture onto the base sound.

5.3.4 Imposition process

Although the most direct adaptation of the CNN-based visual texture synthesis might be to synthesize a log-spectrogram, then decompress and invert it using approximation methods such as the Griffin-Lim algorithm, such a process is hazardous when working with synthesized spectrograms. Additionally to the risk of artefacts being present, the consistency of spectrograms (in the sense given in Section 2.1.2) is also an issue: because the spectrograms needing to be inverted would be synthesized, there is no guarantee that they would correspond to an existing time signal. In turn, this increases the risks of artefacts in the inverted time signal.

[16] H lost its Gram title in the process, since we are not performing the products between arrays of vectors.

Because of this, and given our similar approach with the perceptually-based synthesis presented in [McDermott and Simoncelli 2011] and detailed in the previous chapter, we perform the imposition of the parameters directly on the base audio signal. Such time domain synthesis is also found both in [Tomczak et al. 2018] and [Barry et al. 2018]. Because the imposition of the parameters is directly performed on the time signal, this fundamentally prevents the algorithm from synthesizing a non-consistent spectrogram.

The computation of the gradient of the texture loss can easily be performed using the method presented in [Caracalla et al. 2017]. In practice however, it is similarly computed using the tensorflow[17] library: because tensorflow is optimized for deep learning computations using Graphics Processing Units (GPUs), it also allows for overall much faster computation times than our implementation.

Inspired by both [L. Gatys et al. 2015] and [Ustyuzhaninov et al. 2016], and given the poor results obtained in early attempts using the standard gradient descent algorithm, we opt for the use of the L-BFGS optimization algorithm (briefly described in Section 2.3.4). This algorithm also presents the advantage of not needing a specified learning rate, and was left with its default parameters in the scipy[18] python library (beside its early stopping criteria being nullified), interfaced with the tensorflow library. Although stochastic optimization algorithms like Adam (see Section 2.3.4) could be used instead, they are specifically designed for the training of neural networks on large data-sets: they aim at approximating the true gradient of a loss function over the whole data-set using only batches of it, and as such are not adapted to the optimization of a single signal (as is the case in our algorithm).

5.3.5 Overview of the synthesis

An overview of the algorithm is presented in Figure 5.8. Its global organization resemble that of the CNN-based visual synthesis algorithm, and our contributions can be summarized as:

- Using the compressed log-spectrogram as equivalent to images.
- Parametrizing textures in a non-invariant way along the frequency axis.
- Performing the parameter imposition directly on the audio signal.

5.3.6 Early presentation of the results

Results obtained using this method can be found at `http://recherche.ircam.fr/anasyn/caracalla/thesis/spec_ref.php`. They are analyzed and discussed in depth in the next chapter.

[17] See `https://www.tensorflow.org`
[18] See `https://www.scipy.org`

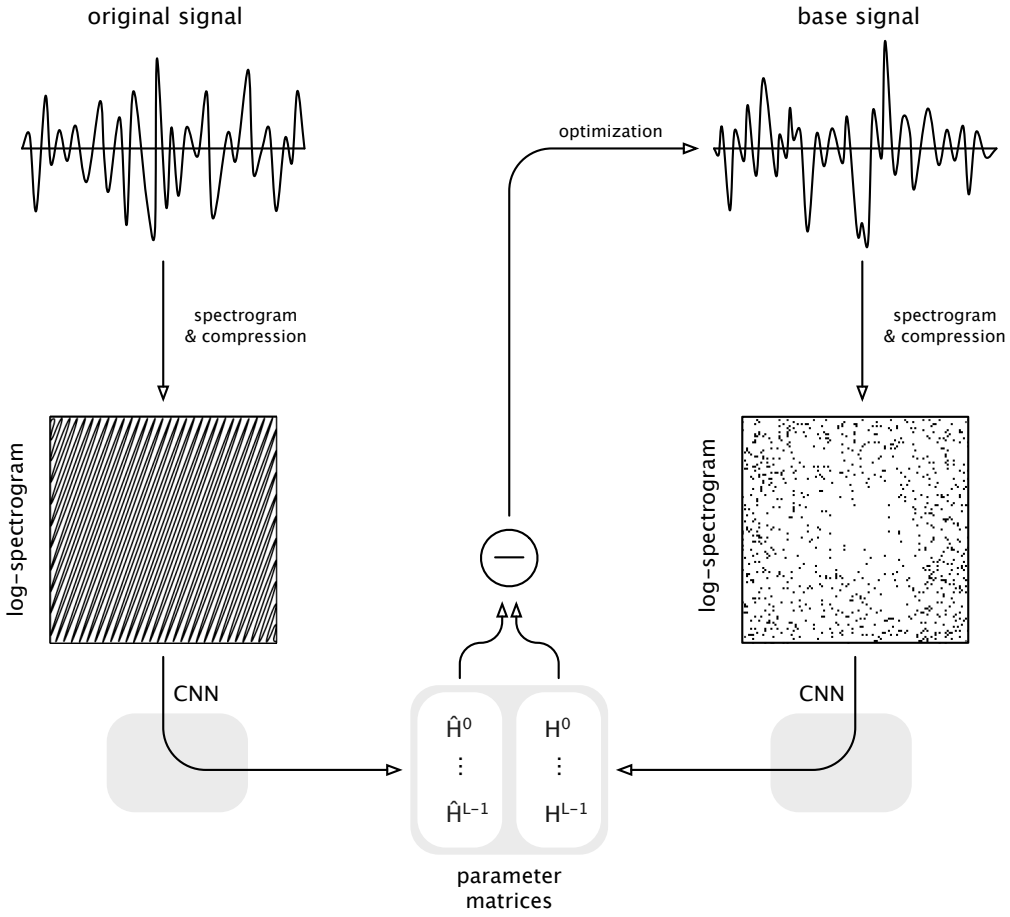


Figure 5.8: *Imposition of the statistics from the original sound texture onto a base signal. The log-spectrogram of each are computed, then passed through the same CNN, and for both of them the cross-correlations between the feature maps of each layer are stored inside sets of parameter matrices. The base signal is then iteratively modified until its set of parameters resemble that of the original texture.*

5.3.7 Other attempts

Two works similarly inspired by [L. Gatys et al. 2015] were published around the same time as ours, although both take a slightly different approach to it. The first is presented in [Ulyanov et al. n.d.], and is the most straightforward application of [L. Gatys et al. 2015].

Method of Ulyanov & Lebedev

This method uses the spectrograms of sounds sampled at 22050 Hz with a 2048 window and 512 hop-size as representation but, instead of using them as a 2D images, uses the frequency dimension as depth (like colors in visual CNNs). This means that instead of convolving filters with the input of the network along both time and frequency axes, the convolution is performed only over the time axis with filters systematically spanning all frequencies.

The CNN chosen is a randomly initialized one-layer CNN with 4094 filters of sizes $(1, 11)$ and stride of $(1, 1)$. The parameters chosen are the Gram matrices as described in Eq. 5.1: because frequency is used as depth, this parametrization is not

frequency-invariant, hence why Gram matrices are suited as parameters. It may also be noted that since the network is only one layer deep, using the frequency as depth is strictly equivalent to using spectrograms as 2D images but with filters having the same height as the spectrograms.

Synthesized spectograms are then inverted using the Griffin-Lim algorithm to produce a time signal. Examples of such synthesized sounds are available at `https://dmitryulyanov.github.io/audio-texture-synthesis-and-style-transfer/`. Because most of those examples poorly qualify as sound textures, it is hard to judge the quality of this synthesis method. However, from the presented sounds one can hear that chirping artefacts are present during impact synthesis, and that synthesized sounds are overall noisy and mildly satisfying. Additionally, rhythms present in the original texture are not well reproduced.

Method of Antognini & al.

The other such method is presented in [Antognini et al. 2018], and may be seen as an extension to [Ulyanov et al. n.d.].

It uses the same spectrogram representation, albeit with sounds sampled at 16000 Hz, windows of 512 samples and a hop-size of 64, with frequency as depth. Instead of using a single CNN with one random convolutional layer, it uses 6: each one possesses 512 filters of size $(1, X)$ with X a power of two ranging from 2 to 64. Just like in the previous method, synthesized spectrograms are also inverted using the Griffin-Lim algorithm to output the resulting audio signal.

Its main departure with the method of [Ulyanov et al. n.d.] lies in its use of other parameters in addition to the Gram matrices. Those parameters are the auto-correlations of the feature maps for shifts ranging from 200 ms to 2 s and are aimed at capturing the rhythmic patterns that the method of [Ulyanov et al. n.d.] fails to reproduce. After noticing that using those two sets of parameters over-constrains the synthesis and results in an exact copy of the target texture, the authors propose the use of an added component to the texture loss. This diversity loss is expressed as:

$$\mathcal{L}_{div} = \max_{\tau} \left(\frac{\sum_{l,i,n} (\hat{F}^l_{in})^2}{\sum_{l,i,n} (F^l_{i(n+\tau)} - \hat{F}^l_{in})^2} \right) \tag{5.7}$$

with F^l_{in} the element at position n[19] of the feature map i of the network l, and the hat denoting the feature map of the target texture.

The final texture loss is then defined as:

$$\mathcal{L} = \mathcal{L}_{gram} + \alpha \mathcal{L}_{auto} + \beta \mathcal{L}_{div} \tag{5.8}$$

with $\mathcal{L}_{gram}$ the distance between the Grame matrices of the target texture and those of the base sound, $\mathcal{L}_{auto}$ the distance between their auto-correlation parameters and $\mathcal{L}_{div}$ the diversity loss expressed above. α and β are used to tune the different contributions and may need to be modified depending on the target texture.

[19] Since the frequency dimension is used as depth, the feature maps of those CNN are 1D and only need one index.

An extensive array of synthesized sounds is available at `https://antognini-google.github.io/audio_textures/baselines.html`, where one can find comparisons between results obtained using this method, that of [Ulyanov et al. n.d.] and that of [McDermott and Simoncelli 2011][20]. Results obtained using the method of [Antognini et al. 2018] are overall more satisfying than those obtained by Ulyanov & Lebedev: they succeed in reproducing the rhythm of the original and are less noisy.

[20] Please note that for reasons unknown, and while all other sounds are presented at a sampling rate of 20 kHz, sounds synthesized using Antognini & al.'s method are presented at 16 kHz: this often affects their perception negatively.

References for chapter 5

Antognini, J., Hoffman, M., and Weiss, R. J. (2018). "Synthesizing diverse, high-quality audio textures". In: *arXiv preprint arXiv:1806.08002* (cit. on pp. 43, 84, 85, 88, 92, 96, 100, 106–108, 112, 123).

Barry, S. and Kim, Y. (2018). *"Style" Transfer for Musical Audio Using Multiple Time-Frequency Representations* (cit. on pp. 43, 82).

Caracalla, H. and Roebel, A. (2017). "Gradient conversion between time and frequency domains using wirtinger calculus". In: *Digital Audio Effects (DAFx)* (cit. on pp. 62, 82, 122).

– (2019). "Sound texture synthesis using convolutional neural networks". In: *Digital Audio Effects (DAFx)* (cit. on pp. 79, 123).

Erhan, D. et al. (2009). "Visualizing higher-layer features of a deep network". In: *University of Montreal* (cit. on p. 74).

Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). "Image style transfer using convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (cit. on pp. 42, 70, 124).

Gatys, L., Ecker, A. S., and Bethge, M. (2015). "Texture synthesis using convolutional neural networks". In: *Advances in neural information processing systems* (cit. on pp. 42, 48, 70, 72–74, 76, 79, 82, 83, 90, 92, 101, 123, 125).

Luo, W. et al. (2016). "Understanding the effective receptive field in deep convolutional neural networks". In: *Advances in neural information processing systems* (cit. on pp. 75, 76).

McDermott, J. H. and Simoncelli, E. P. (2011). "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis". In: *Neuron* (cit. on pp. 42, 48, 49, 52, 54, 55, 60, 62, 63, 65, 71, 82, 85, 88, 107, 108, 112, 115, 122, 123).

Portilla, J. and Simoncelli, E. P. (2000). "A parametric texture model based on joint statistics of complex wavelet coefficients". In: *International journal of computer vision* (cit. on pp. 42, 50, 70, 71, 73, 123).

Simonyan, K. and Zisserman, A. (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (cit. on p. 70).

Tomczak, M., Southall, C., and Hockman, J. (2018). "Audio Style Transfer with Rhythmic Constraints". In: *Digital Audio Effects (DAFx)* (cit. on pp. 43, 82).

Ulyanov, D. and Lebedev, V. (n.d.). *Audio texture synthesis and style transfer* (cit. on pp. 43, 83–85, 88, 91, 92, 107, 108, 112, 123).

Ustyuzhaninov, I. et al. (2016). "Texture synthesis using shallow convolutional networks with random filters". In: *arXiv preprint arXiv:1606.00021* (cit. on pp. 79, 80, 82, 90–92).

Chapter 6

Update to the CNN-based synthesis

Chapter overview

The synthesis of impacts and high correlations across frequency bands is identified as the main weakness of our CNN-based sound texture synthesis. After experimenting with various properties of the CNN used in it, such as its training and the shape of its filters, the time-frequency representation used is investigated. The log-spectrogram representation is swapped for a real and imaginary part (RI) representation of the STFT, and the resulting synthesis algorithm is thoroughly detailed.

Contents

Our method for sound texture synthesis, introduced in the previous chapter, leads to mildly satisfying results. While our intuitions when designing the algorithm seem reasonably founded, the quality of the textures synthesized by this algorithm is not convincing enough: furthermore, this algorithm does not perform notably better than the method introduced in [McDermott and Simoncelli 2011], nor does it particularly distinguish itself from other CNN-based methods such as those presented in [Ulyanov et al. n.d.] and [Antognini et al. 2018].

After obtaining these results and identifying their weaknesses, we performed a batch of tests to both confirm the hypotheses we had made during the designing of our algorithm and identify ways to substantially improve it: the following sections are a summary of these tests and of the insight we drew from them.

6.1 Discussion of the results

In order to properly judge the results of our synthesis algorithm, we used an array of textures presenting characteristics as varied as possible: from simple monotonous textures (bees buzzing, blender noises) to textures possessing salient events, both rhythmic (birds chirping regularly) and random (fire cracking). We also used human noises (crowd hubbub) and various environmental sounds (wind howling, water lapping or flowing). In addition to those, we occasionally used non-textural sounds to highlight some properties of the algorithm.

6.1.1 Impacts re-synthesis

When comparing the original and synthesized texture of sounds such as fire or applause recordings (available at `http://recherche.ircam.fr/anasyn/caracalla/thesis/spec_ref.php`), it appears that our algorithm poorly recreates impacts. This is visible on the spectrograms presented in Figure 6.1: the impacts, recognizable as tall vertical lines on the original spectrogram, are not well reproduced on the spectrogram of the recording. In their stead, segments that span only portions of the frequency axis are found: it may however be noted that the width of those lines appears to be identical between the original and synthesized sounds. This results in cracks being replaced by watery sounds, which also have smaller temporal amplitudes since not all frequencies are excited at the same time.

While this defect in our algorithm is easily noticeable when synthesizing simple impacts, like those of the fire cracking, it extends to all sharp attacks. While the sound of the blender texture first appears rather well reproduced, a closer inspection reveals that the frequent and rhythmic attacks found in the original are drowned in the synthesis: the resulting texture is devoid of them, losing its harshness in the process.

This means that, much like the algorithm of [McDermott and Simoncelli 2011], ours cannot recreate the strong and simultaneous correlation between bands that occurs during impacts. This can be illustrated by re-synthesizing spoken voice signals, despite them not being textures. Due to the presence of a great number of harmonics in human voice, those signals are extremely correlated frequency-wise: when attempting to re-synthesize one, it is noticeable that that despite being locally

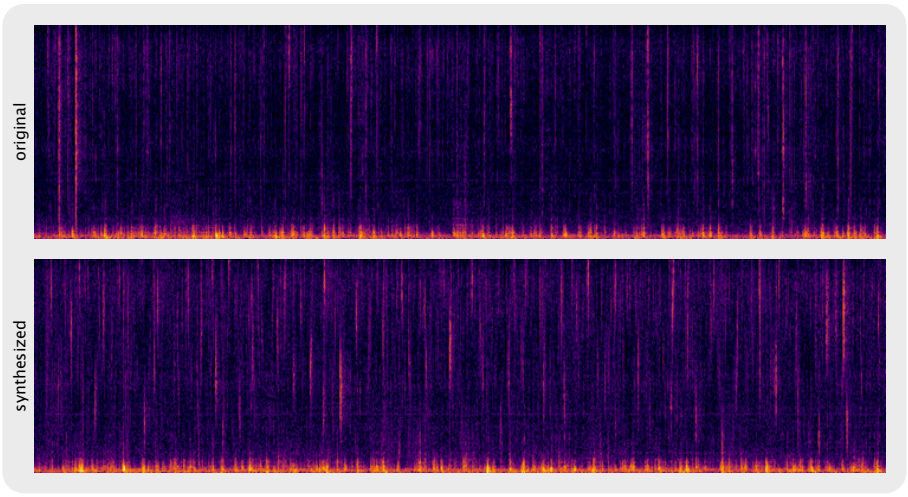


Figure 6.1: *Visual comparison between the log-spectrograms of a fire texture and the texture synthesized by our algorithm.*

correlated, harmonics that are far apart in original spectrogram are not synchronized in the synthesized one. This phenomenon is visible on Figure 6.2, and results in higher harmonics singing “by themselves” (such a signal is available among our synthesized samples under the name of "singing"). Given that our parametrization is implicitly based on pattern description, this implies that the network we use is not adapted to the description of tall patterns.

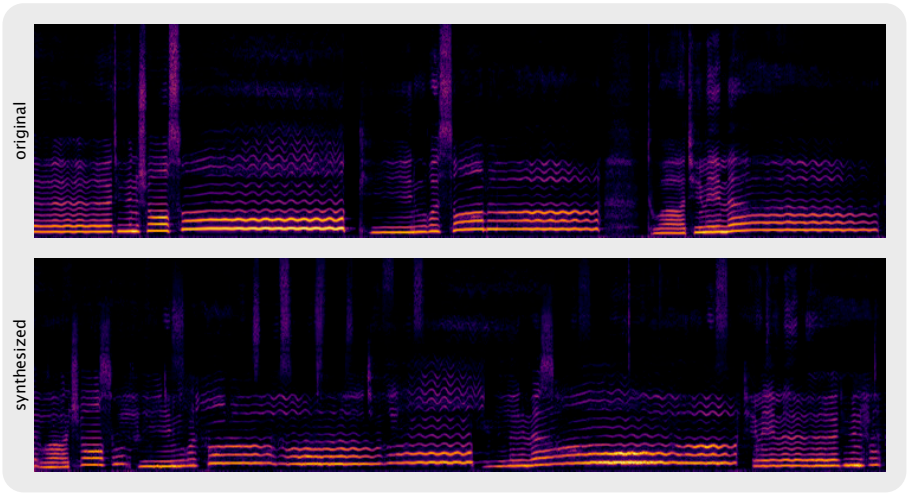


Figure 6.2: *Visual comparison between the log-spectrograms of an singing voice signal and the signal synthesized by our algorithm.*

6.1.2 Border effect

Although it is harder to detect on some of them, all synthesized textures present a border effect at the temporal start and end of the signal: they are extremely similar to the original in the first and last frames of their spectrograms. This implies that despite the cross-correlations being averaged in our set of parameters, an amount of spatial information is still being conveyed by them.

This effect is also present in [L. Gatys et al. 2015], in which the authors notice that the yellow box at the bottom left of the target image is always re-synthesized near its original position (this effect is visible in Figure 5.2): they suggest that this might be due to the use of zero-padding in the convolutional layers. Textures shown in [Ustyuzhaninov et al. 2016] also possess this border effect.

In our synthesized spectrograms, this effects appears to last a number of frames corresponding to the horizontal size of the biggest filter of the network. As such it has no influence over most of the texture, and was thus not highly prioritized.

6.2 Investigation on CNN architecture

We first conducted a series of test on the role of the CNN architecture, aimed at both confirming and developing our understanding of the algorithm as well as correcting its defects. To do so we proceeded by first choosing an architecture as reference. For practical reasons of computation times, we opted for a lighter version of the architecture presented in Section 5.3.2: our only modification to it was the reduction of the number of filters from 128 to 32 per filter shape. The effect of this reduction is notably discussed in the following section. Given this reference architecture, we only modified one of its characteristics at a time so as to isolate the consequences of this modification. If not specified, all parameters of the following network are thus identical to our reference architecture. When modifying only one characteristic of the network was impossible (for instance when investigating a completely different architecture), we tried to have both the reference and the investigated architecture be as equivalent as possible: both in parameter numbers and in the ERF size of their respective filters.

6.2.1 On the influence of filters number

The most straightforward modification of our network with the aim of improving its results is to modify the number of filters in its convolutional layer. In order to test the consequences of such a change, we compared our reference network with two others: one with 128 filters per filter shape (thus having the same architecture as in Section 5.3.2), and one with 4. The results obtained with one and the other are available for comparison at `http://recherche.ircam.fr/anasyn/caracalla/thesis/up_filt_num.php`.

The reduction of the number of filters is expectedly harmful to the quality of the synthesis, and can be interpreted as the parametrization being able to encode a smaller array of patterns. However, increasing this number seems to not have much of an impact. Texture synthesized using more filters seem to be perceptually similar to those synthesized using the reference network. Further tests showed this to stay true despite increasing the number of filter even more: this means that the synthesis abilities of a given architecture cannot be indefinitely improved by increasing the number of its filters. Additionally, this also validates our choice of using a lighter version of the architecture presented in Section 5.3.2 as reference.

6.2.2 On the need for network training

Our choice of working with a random network was initially motivated by [Ustyuzhaninov et al. 2016], the observations made in [Ulyanov et al. n.d.] indicating that trained and untrained network performed similarly, and our understanding that using a random network with enough filters would still allow all possible patterns to be described. So as to verify this, we trained a CNN on a multi-class sound classification task: our aim in doing so was to train the filters of this network and have them specialize in recognizing common spectrogram patterns, then verify if this specialization improved the results of the synthesis.

The architecture of this network consists in the alternation of 4 convolutional layers, each with 64 filters of size (3,3), strides of (1, 1) and ReLU activation function, and of 3 average pooling layers with both a pool size and a stride of (2, 2). A flattening operation is performed on the feature maps of the last convolutional layer, and is followed by a single dense layer with softmax action. The number of neurons is equal to the number of sound classes in our dataset. This design choice was made following our goal of having equal numbers of parameters and comparable ERF between this architecture and the reference one.

The sounds used for the training of the network were extracted from the collaborative Freesound dataset[21] using its API, and were gathered into 8 distinct categories: acoustic music, bird-songs, flowing water recordings, insects recordings, electronic music, rain recordings, traffic recordings and spoken voice recordings. These categories were mainly chosen to represent a wide array of sounds containing both noisy, harmonic and percussive elements, while being adapted to the array of sounds available in the Freesound dataset. Each of those classes contained about 1 total hour of recordings.

Results obtained using this trained CNN are available at `http://recherche.ircam.fr/anasyn/caracalla/thesis/up_net_train.php`: overall, they present no apparent difference with those obtained using the reference network and are even slightly noisier. Be it when visualizing their spectrograms or when listening to them, they present the same properties and flaws as our previous results.

This observation can be further substantiated using a method taken from [Ustyuzhaninov et al. 2016], which aims at comparing the analytical properties of both trained and untrained parametrizations. 4 different recordings of textures (birds singing, brook flowing, crowd hubbub and fire cracking) are broken down into 4 excerpts each, and the distances between the parameters of all excerpts are stored in a similarity matrix: this process is performed both for the trained network and the random reference one. These matrices are displayed in Figure 6.3. The origin of those matrices are at the top left, with the horizontal and vertical axes representing the index of the texture excerpt. Close distances are associated with lighter colors, although the two matrices do not use the same color scale.

[21] See `https://freesound.org/`

The apparent diagonals on the similarity matrices are expected, since the distance between the parameters of an excerpt and themselves is null. The light blocks along the this diagonal are also not surprising: parameter-wise, excerpts of a same texture are closer to each other than to excerpts of other textures. The fact that these blocks are apparent is thus a tell that a parametrization is efficient for texture analysis. This experiment is however not a proof of how suited for synthesis a parametrization is: it only shows that textures that sound similar are close together in the parameter space, not that two sounds close would sound similar.

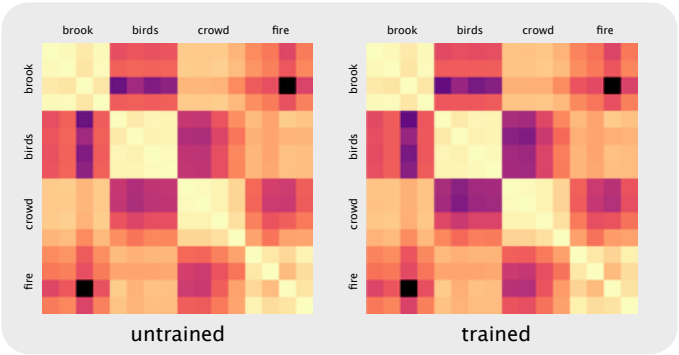


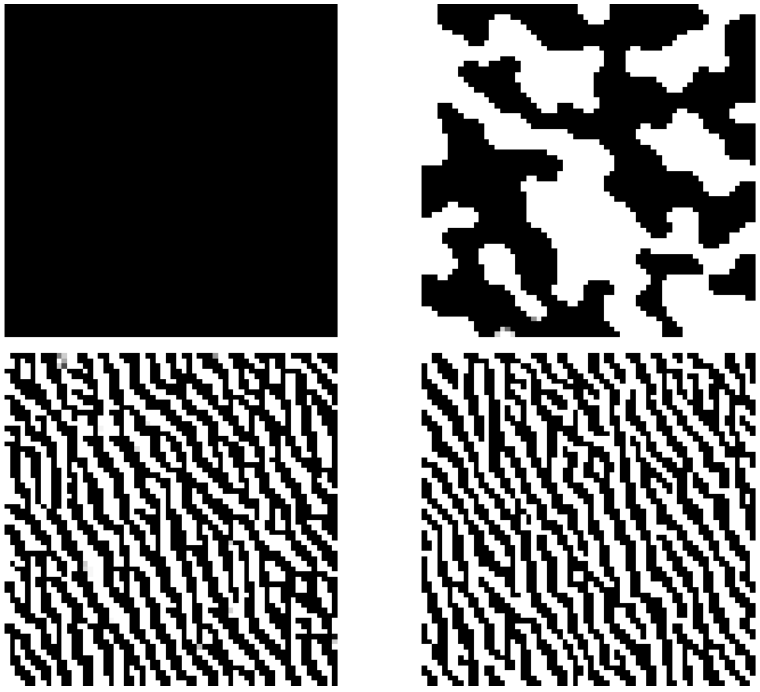
Figure 6.3: *Similarity matrices of an untrained and trained network: this matrices displays the distance between parameters of several textures excerpts. Values are shown using a logarithmic scale.*

The main argument in favor of using a trained CNN, and one used in [L. Gatys et al. 2015], is that the specialization of its filters make its parametrization more pertinent then that of an untrained CNN. This should translate to the similarity matrices displayed in Figure 6.3, where the contrast between similar and dissimilar textures should be sharper for the trained CNN. This is however not the case: even using a logarithmic scale, the similarity matrix of the trained network does not appear to be more discriminating than that of the untrained reference network. This further convinces us that the training of the network is not necessary to create a fitting parametrization of textures.

6.2.3 On the necessity of activation functions

Although ReLU are used in [L. Gatys et al. 2015], [Ustyuzhaninov et al. 2016], [Ulyanov et al. n.d.] and [Antognini et al. 2018], to the extent of our knowledge their role has neither been focused on or explained yet. So as to investigate it, we initially synthesized the identikits of a random CNN with a single convolutional layer and filters of size 3×3, both with and without ReLU. The results of two of those syntheses are shown in Figure 6.4.

The two sets of identikits are extremely similar, beside the fact that removing the ReLU leads to the local appearance of inverted patterns (visible when comparing the identikits at the top of Figure 6.4). This is to be expected since maximizing the function (5.3) may be done by having the feature map reach both high positive values and high negative values. Because ReLU set all negative values to 0, identikits of filter that use them cannot contain strong inverted patterns: hence why they only



(a) (b)

Figure 6.4: *Identikits of the same two* 3×3 *filters of a random one-layer CNN. In (a): with ReLU. In (b): without ReLU.*

appear when removing the activation function.

Outside of the presence of inverted patterns, the identikits are similar both when using ReLU and when not using them: it might thus be expected that they are not necessary to the synthesis, and that similar results may be obtained without using activation functions. To test this hypothesis, we removed all activation function from the reference architecture and used the resulting network to synthesize a set of textures. Audio results are available at `http://recherche.ircam.fr/anasyn/caracalla/thesis/up_relu.php`. The comparison of the spectrograms of an original texture, the texture synthesized from it with the reference network and the texture synthesized from it by the ReLU-less reference is shown in Figure 6.5. Contrarily to what we were expecting, those results are systematically more noisy and and more blurry (visually) then the references: this lends weight to the idea that the ReLU (or potentially any activation function) is necessary to CNN-based parametric synthesis. As of yet, we have not reached any satisfactory explanation of this fact and have temporary settled on using ReLU in all our architectures.

6.2.4 On the influence of filter shapes

As detailed in Section 5.2.2, it is our understanding that only patterns of the same size as the filters may be encoded in the parametrization of our algorithm. To further confirm this interpretation, we performed batches of textures synthesis using networks with filters of different sizes.

A first batch of synthesis was performed with square filters of 6 times the size of those of the reference architecture, and another with square filters 3 times smaller

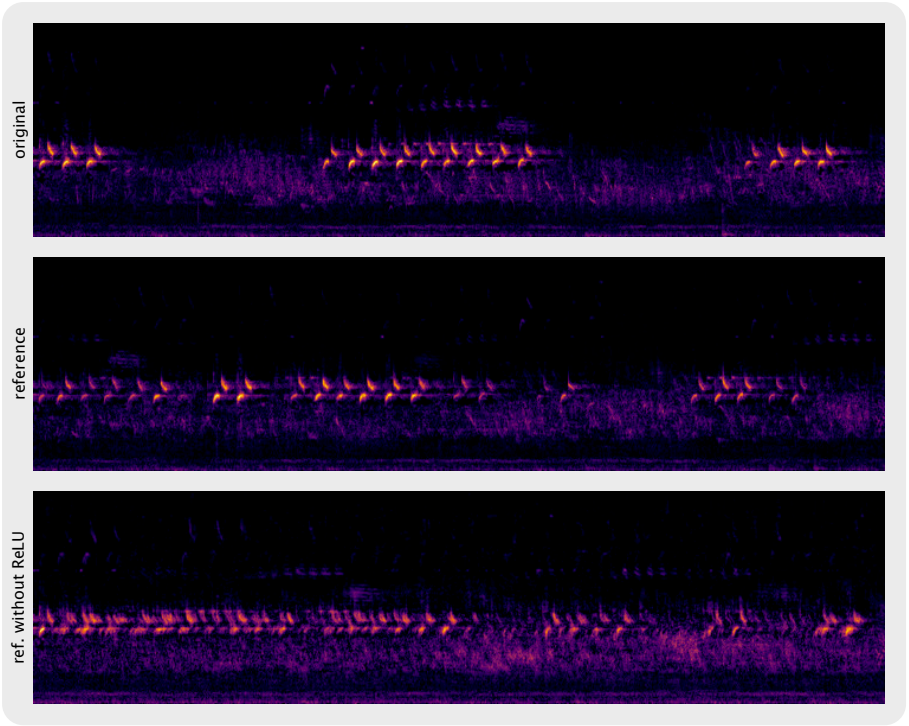


Figure 6.5: *Visual comparison between the log-spectrograms of a singing birds texture, the texture synthesized by the reference architecture and the texture synthesized by the reference architecture with its ReLU removed.*

than the reference. Their results are available at `http://recherche.ircam.fr/anasyn/caracalla/thesis/up_filt_size.php`, while the comparison between synthesized fire textures is presented in Figure **??**. As visible on the spectrograms, textures synthesized using small filters present very little organization, even at a medium scale. On the other hand, those synthesized using large filters are organized very similarly to the originals, even managing to better recreate impacts.

While this would be an argument in favor of using larger filters in our network, listening the synthesized sounds immediately reveals that textures synthesized using big filters are noisier than our reference results. Although this may come as a surprise, this phenomenon can be explained as follows: by using the same number of filters but increasing their sizes, our selection of filters covers a much more reduced part of the ensemble of possible filters. Indirectly, this results in a deterioration of the descriptive abilities of the parametrization. Because of this, it is not advised to simply use filters as large as possible: instead, it is better to design them so that they are just large enough to enforce local correlations at a given scale. Additionally, it is important to chose a number of filter that is coherent with their sizes. It is also important to keep in mind that enforcing correlations on a very large scale increases the risks of the target texture being recreated perfectly: by imposing patterns that are as large as the target texture, its structure will be exactly reproduced in the synthesized texture.

This knowledge can be put to use to solve the issue of impact re-synthesis. Instead of using large square filters, it is possible to use tall filters to enforce long-distance

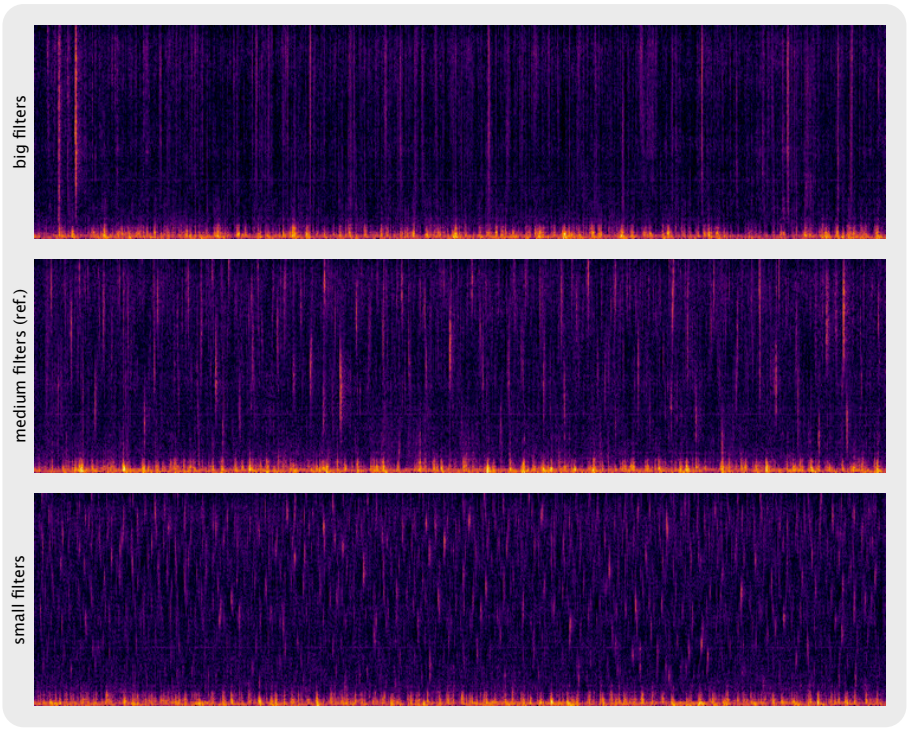


Figure 6.6: *Visual comparison between 3 fire texture log-spectrograms synthesized using filters of different sizes.*

correlations across the frequency axis. Despite being tall frequency-wise, these filters can be chosen as slim as possible to keep the number of filter weight as small as desired. This method was put to use in be designing a network with 128 filters of each of the sizes [(3, 3), (11, 3), (53, 3), (101, 3), (11, 11), (31, 11), (53, 11), (27, 27)]. The results of syntheses performed using this network are available at `http://recherche.ircam.fr/anasyn/caracalla/thesis/up_tall.php`, while the synthesized log-spectrogram of the same fire texture as Figure 6.1 is displayed in Figure 6.7. Both when looking at this spectrogram and listening to the audio results, impacts appear at first to be recreated in a more convincing fashion. However, this reconstitution is not perfect either: while spectrograms of the synthesized sounds show that all frequency bands are indeed activated simultaneously, the audio results still lack sharpness and sound too "soft" and watery. Additionally, this also comes to the price of the apparition of a constant noise throughout the synthesized textures: despite trying different sets of filter shapes, we did not manage to select one that both imposed impacts more convincingly and was rid of this noise.

6.2.5 On the influence of network depth

Since ERF increases both via filter size and layer depth, it was our understanding that using a deep random network would yield the same results as using the reference one. To confirm this hypothesis, we used a deep network build identically to the one used to test the influence of training (and which architecture is described in Section 6.2.2), although without training it. Texture synthesized using this architecture are available at `http://recherche.ircam.fr/anasyn/caracalla/thesis/up_deep.php`. It appears that they are indeed of a quality equivalent to that of textures synthesized by the reference architecture, thus confirming our hypothesis. However, manipulating

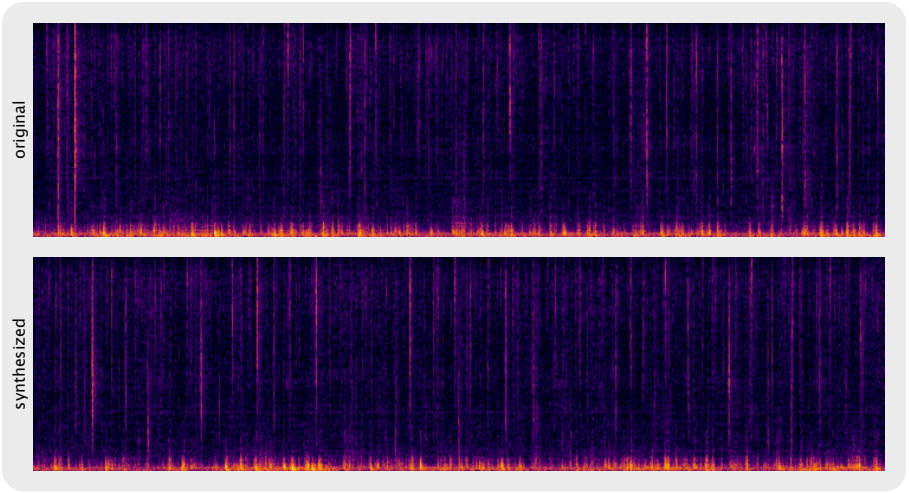


Figure 6.7: *Visual comparison between the log-spectrograms of a fire texture and the texture synthesized by a network containing tall and slim filters.*

networks made of a single layer is more straight-forward: we consequently kept using mostly single-layer networks in our tests.

6.2.6 On the use of multiple CNNs

In [Antognini et al. 2018], the authors replace the random one-layer CNN with filters of different sizes by several random one-layer CNNs to compute the Gram matrices. These CNNs each have filters of a single given shape: in essence, this is equivalent to having one single CNN with filters of various sizes while only considering cross-correlations between filters of identical shape in the parametrization. To test the influence of this design choice, we reproduced it on our reference architecture by discarding those cross-correlations from the parametrization: syntheses using this method are available at `http://recherche.ircam.fr/anasyn/caracalla/thesis/up_multi.php`.

A comparison with results obtained using the reference architecture reveals no noticeable difference between the two, although the multiple CNNs approach drastically reduces de number of parameters (and thus the computation times). This may imply that patterns described by the cross-correlation between filters of different sizes are of lesser importance than those described by cross-correlations between filters of identical size. We initially believed that patterns described by bigger filters might dominate those of smaller filters within the correlation value, and thus that using those cross-correlations as parameter would prove redundant. However if this were true the identikits of their correlation with smaller filters would strongly resemble their own identikits: this is not the case. As with the role of ReLU in our synthesis, we have not yet reached yet a satisfactory explanation of this fact.

6.3 Investigation on representation

Although the different manipulations of the architecture of our network allowed us to better understand the process at work during the synthesis, each of our attempts at increasing the quality of impact synthesis (and more broadly the synthesis of events

with high vertical correlation) resulted in the apparition of a noticeable foreground noise. Despite being more convincing, these impacts also still lack the sharpness of those present in the original textures. However, and as visible in Figure 6.7, the log-spectrograms of these synthesized textures are visually extremely similar to those of the original textures: it is thus probable that the algorithm is working as intended, and that the issue lies rather within the representation used. Because of this, we experimented with different representations in order to get rid of the noise present in synthesized sounds.

6.3.1 Early attempts at alternative representations

Our first three attempts were meant as slight variations on the log-spectrogram representation that we had been using up until this point, and are described in the following sections.

Multi-scale representation

This representation is inspired by works such as [Snelgrove 2017] and aims at giving a view of several scales of the input to the CNN used for the parametrization. To do so we use 4 CNNs: while the first is given the log-spectrogram of the signal, the three others are given down-sampled versions of it. The down-sampling is performed by adding average-pooling layers of respective sizes $(2, 2)$, $(4, 4)$ and $(8, 8)$ upstream of those networks. The CNNs are all random and consist in a single convolutional layer with 128 filters of size $(3, 3)$[22] each.

Multi-resolution representation

The previous representation does not alter the information given to the network, since the different representations used in it are simply down-sampled versions of each others: it is thus to be expected that the results it produces would not differ from those obtained using the reference configuration. So as to use representations that carry different information, it is possible to use a multi-resolution one instead. To do so we simply use 3 log-spectrograms with different STFT parameters each: their window lengths are of 256, 512 and 1024 with hop-sizes being half the length. Each of those representations is then passed through a CNN identical to the reference.

Mel representation

It is also possible to replace the log-spectrogram representation by a Mel-spectrograms one (as mentioned in Section 2.1.3): this may be interpreted as an attempt to spread the imposition process perceptually, and is fundamentally equivalent to weighting the gradient of the texture loss. For this we use 64 triangle-shaped filters spread from 20 Hz to half the sampling rate, with the resulting representation being processed through the reference CNN.

Paradigm failure

The test of those three representations were unequivocal: all synthesized textures were sensibly similar to those obtained by using log-spectrograms: even when using various network architectures they lacked sharpness in their impacts and presented the same noisiness. This implies that the change needed to improve our synthesis method is

[22] Note that this size was chosen so that the biggest ERF of those network is of size $(24, 24)$ and comparable to the ERF of $(27, 27)$ of the reference architecture.

more fundamental than simply weighting or scaling differently the time-frequency representation.

The most obvious flaw of these representation is that they completely disregard the phase of the signal. Although we initially believed that imposing statistics directly onto the time signal implicitly produces an appropriate phase corresponding the spectrogram, what this process guarantees is merely that the spectrogram of the synthesized sound is consistent: the phase implicitly created can still be a poor choice. This might for instance explain why sharp attacks, despite being correctly reproduced in the spectrogram, still yielded poor audio results. It is for this reason that we investigated means to add the phase information to our representation.

6.3.2 RI representation

Colored images are usually represented as 3D matrices, with each slice (called channel) of the matrix being the projection of the image onto each of the 3 primary colors. When using CNNs, the advantage of this representation is that the filters of the first layer are convolved with local chunks of the image across its full depth: as such they are sensible to the local correlation of the different channels. This can hypothetically be transposed to the audio domain by having the magnitude and the phase of a time-frequency representations act as the channels of the input, as can be found in [Engel et al. 2019]. But while correlations across the different color channels of an image are visibly strong (one can usually see that each color channel represents the same image), and while it is understandable that filters may gain from seeking for patterns across them, this is hardly the case when comparing magnitude and phase. This is visible on Figure 6.8 where both the phase and the unwrapped[23] phase have little in common with the magnitude of the STFT: the standard phase looks almost like a white noise, while the unwrapped phase results in horizontal stripes with increased intensity on the right.

However, this is not the only way of decomposing a complex 2D matrix into two real-valued matrices: one can instead simply use the real and imaginary part of this data. When applied to a STFT, this results in two real-valued matrices that are more difficult to physically interpret than the magnitude and phase matrices, but which implicitly contain both. As visible on Figure 6.9 those two representations are strongly correlated, making their use as color channels much more pertinent. Additionally, both real and imaginary part bear an important resemblance to the magnitude, meaning that our understanding of the working of the synthesis algorithm is unchanged when working with them. It must however be noted that those two matrices are not positive, like the spectrogram is, but rather may take both positive and negative values. This is the reason for our use of a diverging color map in Figure 6.9 that is white for values near 0, blue for strong negative values and red for strong positive values. The compression used to make real and imaginary parts more visually understandable must also take this into account: instead of using the log function we thus switch to using a sigmoid compression, which compresses both high negative and high positive values. The details of this compression are given in the following section.

[23] The unwrapping is a process that removes the phase jumps that occur every 2π and results in a more continuous and easily understandable phase signal.

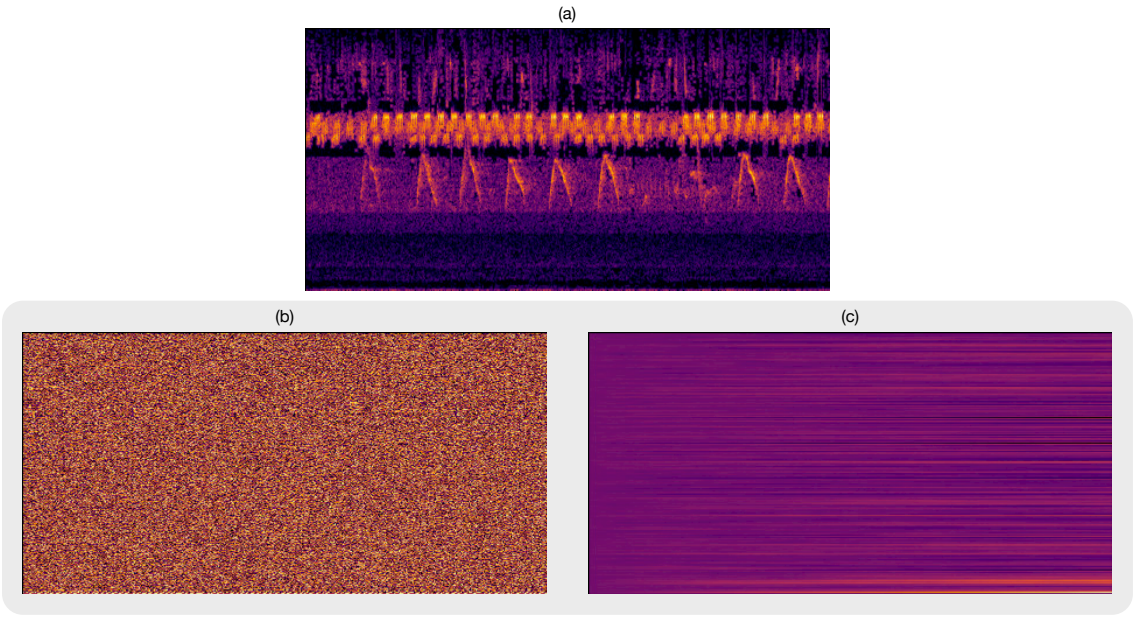


Figure 6.8: *Visual comparison between the magnitude and two phase representations of an STFT: (a) log-spectrogram, (b) phase, (c) unwrapped phase.*

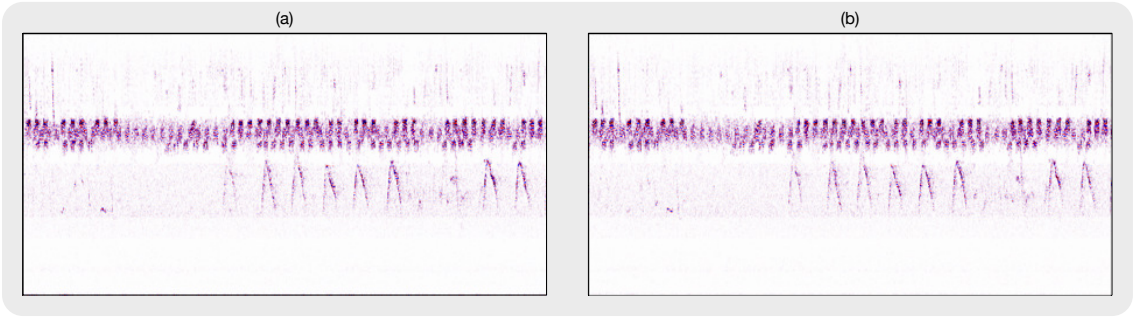


Figure 6.9: *Visual comparison between the real and imaginary parts of an STFT: (a) compressed real part, (b) compressed imaginary part.*

The use of real and imaginary part as channels for a CNN is already present in [Fu et al. 2017] in which they are dubbed RI spectrograms: we thus adopt this name for this representation. Using RI spectrograms as input, we performed an array of syntheses using the reference architecture: the synthesized textures were successful in both containing convincing impacts and being rid of the noisiness that plagued our syntheses up until this point. Combining both this representation and the knowledge gained from experimenting with the network architecture, we then designed the updated version of our synthesis algorithm. This version and its results are thoroughly presented in the following section.

6.4 Presentation of the updated algorithm

This section is a synthesis of our work on sound texture synthesis using CNN-based statistics, and describes in depth the resulting synthesis algorithm. An overview of our contributions is also given at its end.

6.4.1 Representation

The representation that serves as input to the CNN are the compressed RI spectrograms which are organized as color channels to form a 3D matrix. We usually work with sounds sampled at 22050 Hz and use a window length of 512 samples with a hop-size of 256 for the computation of the STFT. Given the STFT X of a sound signal, we first normalize it by the maximum of its absolute value and then compute the RI representation as follow:

$$\begin{cases} R &= 2\sigma(C\,\mathrm{Re}\{X\}) - 1 \\ I &= 2\sigma(C\,\mathrm{Im}\{X\}) - 1 \end{cases} \tag{6.1}$$

with R the compressed real part of the STFT, I its compressed imaginary part, and σ the sigmoid function (as defined in Section 2.3.2). C is a compression factor that we arbitrarily set to 10. Defined this way, both R and I are always comprised between -1 and -1, while being centered around 0.

6.4.2 CNN architecture

Instead of using a single CNN, we use 8 distinct CNNs similarly to [Antognini et al. 2018]. Each is comprised of a single untrained convolutional layer, with 128 filters of one unique size. These 8 sizes of the 8 CNNs are respectively [(101, 2), (53, 3), (11, 5), (3, 3), (5, 5), (11, 11), (19, 19), (27, 27)]. All CNNs also use a stride of (1, 1). In addition to using square filters, we also use tall ones: their aim is to describe thin patterns covering most of the frequency bands, such as impacts. They are also chosen as thin as possible in order to minimize the size of the filters, in accordance with the tests performed in Section 6.2.4. No padding is applied, and all layers include a ReLU activation function. The weights from the filters are drawn from a uniform distribution between 0.05 and 0.05, and no bias is applied.

6.4.3 Parametrization

We use the parameters introduced in Section 5.3.3 as:

$$H^l_{ijm} = \sum_n F^l_{imn} F^l_{jmn} \tag{6.2}$$

with F^l_{imn} the feature map at position (m, n) of the i^{th} filter, but this time of the l^{th} network. As previously mentioned, the same parametrization could be achieved by using a single CNN with several parallel layers while not taking into account the cross-correlations between feature maps of differently shaped filters.

We use the same texture loss, introduced in Section 5.3.3, and defined as:

$$\mathcal{L} = \sum_l \frac{\|\hat{H}^l - H^l\|_2}{\|\hat{H}^l\|_2} \tag{6.3}$$

with H^l the l^{th} parameter tensor with the base sound as input to the network while $\hat{H}^l$ is the l^{th} parameter tensor with the target texture as input. Minimizing this loss is thus equivalent to imposing the parameters of the original texture onto the base sound.

6.4.4 Imposition process

Because synthesizing the real and imaginary part of an STFT does not guarantee that the resulting STFT is consistent, we impose the parameters directly over the time signal (as is done in the previous version of the algorithm). In practice, we use the L-BFGS optimization algorithm and the tensorflow library to perform the iterative imposition of the parameters.

6.4.5 Overview of the synthesis

An overview of the algorithm is presented in Figure 6.10. This algorithm is originally based on the visual texture synthesis algorithm introduced in [L. Gatys et al. 2015], while our contribution can be summarized as:

- Using the compressed real and imaginary part of the STFT as color channels of the same image.
- Parametrizing textures in a non-invariant way along the frequency axis.
- Performing the parameter imposition directly over the audio signal.
- Using multiple CNNs to reduce the size of the parametrizaion while maintaining its efficiency.
- Using long thin filters to better impose sharp and brief impacts.

6.4.6 Presentation of the results

Results obtained using this method can be found at `http://recherche.ircam.fr/anasyn/caracalla/thesis/up_RI.php`. As is audible when listening to them, the algorithm manages to realistically re-synthesize all presented textures: it can convincingly recreate random background events as well as individual salient events (such as in the crickets texture, which contain a singing bird in the foreground), and pitched events as well as impacts (such as in the applause texture). The quality of those synthesized sounds seems overall almost equivalent to that of the originals, and globally higher than that of existing synthesis methods: however, the perceptual comparison of our method with existing ones is the subject of the next chapter.

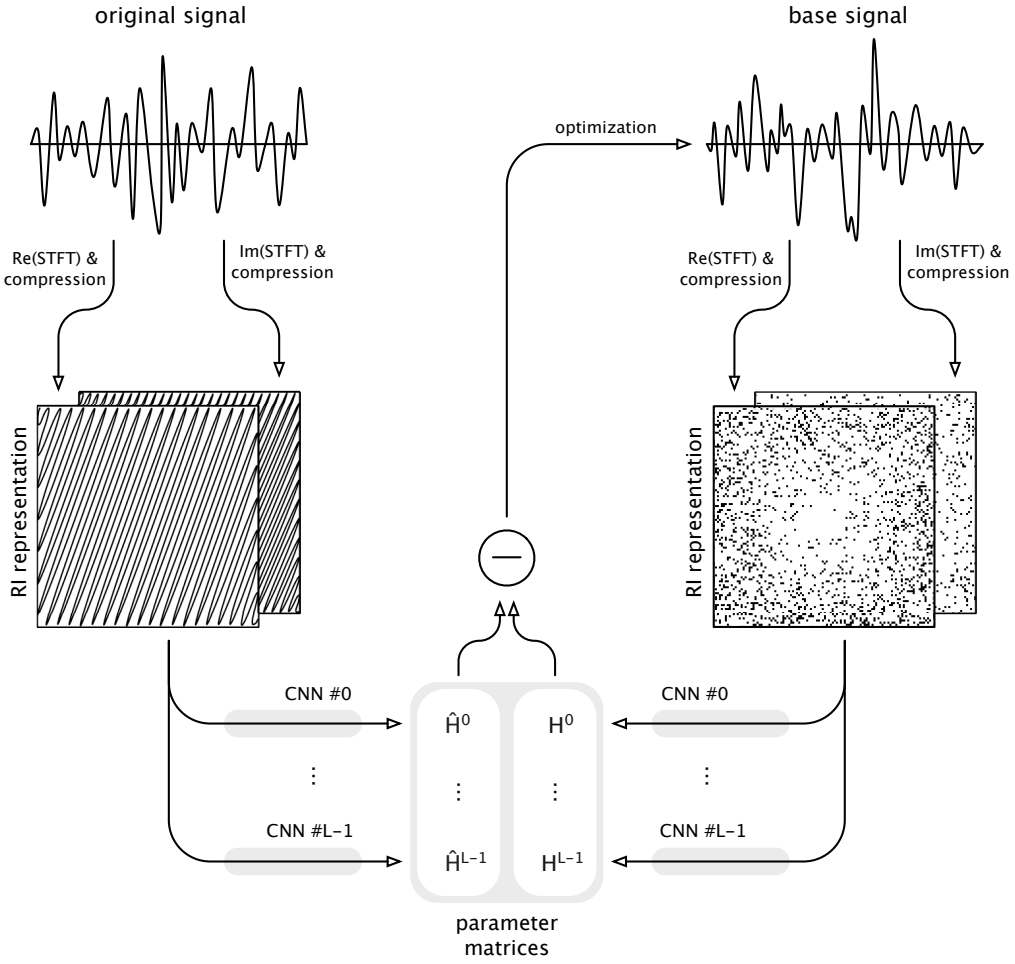


Figure 6.10: *Imposition of the statistics from the original sound texture onto a base signal. The compressed RI representation of each is computed, then passed through the same series of single-layer untrained CNNs. For both original and base signal, the cross-correlations between the feature maps of each CNN are stored inside sets of parameter matrices. The base signal is then iteratively modified until its set of parameters resemble that of the original texture.*

References for chapter 6

Antognini, J., Hoffman, M., and Weiss, R. J. (2018). "Synthesizing diverse, high-quality audio textures". In: *arXiv preprint arXiv:1806.08002* (cit. on pp. 43, 84, 85, 88, 92, 96, 100, 106–108, 112, 123).

Engel, J. et al. (2019). "Gansynth: Adversarial neural audio synthesis". In: *arXiv preprint arXiv:1902.08710* (cit. on p. 98).

Fu, S.-W. et al. (2017). "Complex spectrogram enhancement by convolutional neural network with multi-metrics learning". In: *IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)* (cit. on p. 99).

Gatys, L., Ecker, A. S., and Bethge, M. (2015). "Texture synthesis using convolutional neural networks". In: *Advances in neural information processing systems* (cit. on pp. 42, 48, 70, 72–74, 76, 79, 82, 83, 90, 92, 101, 123, 125).

McDermott, J. H. and Simoncelli, E. P. (2011). "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis". In: *Neuron* (cit. on pp. 42, 48, 49, 52, 54, 55, 60, 62, 63, 65, 71, 82, 85, 88, 107, 108, 112, 115, 122, 123).

Snelgrove, X. (2017). "High-resolution multi-scale neural texture synthesis." In: *SIGGRAPH Asia Technical Briefs* (cit. on p. 97).

Ulyanov, D. and Lebedev, V. (n.d.). *Audio texture synthesis and style transfer* (cit. on pp. 43, 83–85, 88, 91, 92, 107, 108, 112, 123).

Ustyuzhaninov, I. et al. (2016). "Texture synthesis using shallow convolutional networks with random filters". In: *arXiv preprint arXiv:1606.00021* (cit. on pp. 79, 80, 82, 90–92).

Chapter 7

Evaluation

Chapter overview

The two main properties of our synthesis algorithm evaluated in this chapter are its realism and its variability. To evaluate the former, an online evaluation comparing its results to those of other methods is performed: the results of this evaluation are extremely positive, showing our synthesis algorithm to be convincing on a large panel of textures. The variability of the algorithm is also studied and the link between the re-synthesis of salient events and the local similarities between original and synthesized textures is made.

Contents

The goals of our sound texture synthesis algorithm, as stated in the introduction to this thesis, are rather clear in their definition. The aim is to be able to synthesize varied and realistic texture that might be extended and controlled. The meaning of "varied" is twofold: we wish for textures generated from the same texture to not be identical to each others and to the original (variability), but also wish for the synthesis method to work for a wide array of textures (flexibility). But to judge how much an algorithm meets those goals is less straight-forward. It is, however, an important task: in essence, it brings answers to the question "how well does this synthesis algorithm work ?". This evaluation is the focus of the current chapter.

7.1 Motivation

A criterion like the extensibility of a synthesis is easy to evaluate objectively: it is met, or is not. Control, albeit more subjective than extensibility, can also be approximately judged given the range of parameters available when using a synthesis method. Despite not being addressed in length in our work, those two points are discussed in the following chapter. We thus focus on three points: realism, variability and flexibility.

Realism is evaluated in [Antognini et al. 2018] using a VGG-ish score: this score is the Kullback Leibler divergence between the predictions of the VGG-ish CNN[24] for the original and for the synthesized texture. This divergence is a measure of how close two probability distributions are, and is meant as a way to evaluate how resembling the two sounds are to a CNN trained on sound recognition. Although such a quantitative approach is convenient, we believe that it may only be used as a rough estimate to the evaluation of realism. As mentioned in Section 3.2, having access to a quantitative score expressing how close two textures are (without this score being a simple measure of how identical they are) means it could directly be used for texture synthesis: by acting as a texture loss, the score would implicitly contain a texture parametrization. Given that this does not work in practice, the VGG-ish score cannot be used as a meaningful realism score. A perceptual evaluation performed by human listeners is thus the most reliable way of judging the realism of a synthesis.

Flexibility does not need to be directly evaluated, and is rather implicitly contained in the evaluation of realism: if this evaluation is positive on a wide array of textures then the algorithm may be deemed flexible.

As a measure of how different synthesized sounds are from the original (and between themselves), variability may be partly evaluated in an objective and quantitative fashion: although we may for instance need to decide of acceptable levels of resemblance, several distance evaluations are available for use.

7.2 Realism evaluation

It may seem somehow trivial to evaluate whether a sound texture is realistic and resemble its original version: in most cases the differences are clearly audible to

[24] Available at `https://github.com/tensorflow/models/tree/master/research/audioset`

a trained ear. But although we have done so extensively during the course of our work, evaluating synthesized sounds on one's own is hazardous. For instance, quite some time was lost during the tuning of our algorithm because we progressively and involuntarily focused on a given set of artefacts: in quite the same way that long listening sessions tire the ears and burden the work of sound engineer during a mastering session, this resulted in us increasingly ignoring important defects in the algorithm.

This bias can be compensated for by using the judgement of a large amount of listeners. In order to do so, we designed an online (so as to reach the largest possible audience) perceptual test which is detailed in the following sections.

7.2.1 Evaluation preparation

Before discussing the presentation of the test, we begin by clearly defining its goals and design.

Design choices

The core goal of the test is to evaluate in an unbiased way the realism of our synthesis method. To be meaningful, the realism ratings resulting from the test need to be compared to understandable references. Given a synthesized texture, such references can be the realism ratings of a "perfect" and a "bad" synthesis. While a coarse synthesis is easily created, finding a perfect example is more complex. Supposing that the original texture is constant enough in its behavior, we decided to use its continuation as the best possible synthesis. While the original texture could be argued as being perfectly resembling to itself, the realism we evaluate is not simply a measure of how identical the original and the synthesized sounds are: instead, it is a measure of how much the synthesized sound may be said to be of the same kind of texture as the original, as if it had been recorded in the same conditions. As such, using the original as the "perfect" synthesis may be deceiving.

This protocol is loosely based on MUSHRA (Multiple Stimuli with Hidden Reference and Anchor), defined in [ITU-R 2003], in which a hidden reference (previously our "perfect" sample) and an anchor (our "bad" sample) are hidden among the test samples. This methodology is intended for the evaluation of the perceived quality of compression algorithms, and as such participants are asked to evaluate how identical each test sample is to the original: this is not the case in our evaluation, hence why it is only an adaptation of MUSHRA. We however used both a hidden reference and an anchor, as well as the 0-100 scale over which test samples are rated in MUSHRA.

Additionally, this test is an opportunity to compare our latest synthesis method to other parametric methods (notably those presented in [McDermott and Simoncelli 2011], [Ulyanov et al. n.d.] and [Antognini et al. 2018]) but also to the two previous algorithms of our design (presented in Chapters 4 and 5). While it might also be seen as an opportunity to further confirm the effects of the CNN architecture detailed in the previous chapter, early tests have led us to believe that the differences between those versions are poorly perceived. Additionally, we wished for the test to be answerable in a reasonable amount of time: covering several versions of the same method in addition to a wide array of textures and methods thus seemed ill-advised.

For lack of experts in sound textures, we intended this test to be performed both by audio professionals and untrained participants. Consequently, the instructions given to them needed to convey exactly what we wanted them to evaluate. Because sound textures are not commonly discussed, this task is more complex than it may seem: we needed to ask them to evaluate how alike a synthesised texture was to the original, without it being interpreted as a question about how identical they were. Asking for the quality of the synthesis was also risky: most textures are at least partially noisy, and we risked having participants rate how clean the resulting synthesis sounded. Much like how sound textures are more easily defined by example, we decided on using an image and ask them to judge how plausible it would seem that a synthesized sound had been recorded moments after the original.

Although this test could also serve to rate the variability in synthesized sounds, we decided on not doing so. We believe that trying to get across to the participant that we want them to separately evaluate both how close the style of a synthesized sound is to that of the original and how identical the two sounds are would only make our instructions more confused. For this reason, the evaluation of the variability of the synthesis is done separately.

Sample choice and pre-processing

In order to test the flexibility of the different algorithms present in the evaluation, it is necessary that the samples used in the test cover an array of texture as broad as possible. For the test to be answerable in a reasonable amount of time, this means that the samples used need to be both diverse and representative of various texture characteristics. Those characteristics may for instance refer to the origin of the texture (e.g. environmental, human or mechanical) or its content (e.g. stationary noises, impacts, harmonic events).

With the kind consent of Dr. Joseph M. Antognini, we used the set of sounds available at `https://antognini-google.github.io/audio_textures/baselines.html` to choose our original samples from. In addition to containing a wide array of textures, this set also contained the synthesized versions of each texture using the methods presented in [Antognini et al. 2018], [Ulyanov et al. n.d.] and [McDermott and Simoncelli 2011]: this proved an invaluable time gain at a much need period of our work, and for this we extend our sincerest thanks to Dr. Antognini. From this set we chose 10 original textures (within brackets are given their original names in the dataset): applause (Applause2), bees (Bee swarm1), birds (Birds in tropical forest), crowd (Large diner), fire (Fire3), insects (Insects during day in South), rain (Rain beating against window panes), sink (Bathroom sink), static (Radio static1) and wind (Wind - moaning).

All sounds taken from this dataset are sampled at 20 kHz, save for sounds synthesized using the method of Antognini & al. which are sampled at 16 kHz. Because this tends to have a strong negative impact on the perception of those sounds, we chose to down-sample all sounds used in our test to a sample rate of 16 kHz.

All sounds taken from this dataset are 7 seconds long, apart from those synthesized using the method of McDermott & Simoncelli which are 5 seconds long. Given the

choice of 10 textures and the 8 methods to test (our 3 methods, that of Antognini & al., that of Ulyanov & Lebedev, that of McDermott & Simoncelli and both the hidden reference and the anchor), we chose to use samples of 4 seconds. The original textures were presented as the first 4 seconds of the 10 chosen sounds, and the hidden reference as the last 4 seconds. Anchors were created by filtering white noises so that their spectrum had the same distribution as the originals. Sounds synthesized using our methods used the complete 7 seconds-long originals. All synthesized sounds were then evenly cropped to have a length of 4 seconds. All sounds used in the test (including the originals, hidden references and anchors) were windowed by a Tukey window with a ratio of 5% to avoid any discontinuity at the boundaries.

Because uneven audio volumes may influence the perception of artefacts, we normalized all sounds so that their energy (or variance) were identical. This was especially important for anchors, which otherwise had a much louder perceived volume when normalizing each sound by its maximum value.

Evaluation presentation

The most fundamental aspect of the presentation of the evaluation is that it needed to present the task in a clear way while minimizing any bias it might convey to the participants.

The test started with a brief introduction to what sound textures were, and gave a few examples of texture:

> "This test aims at evaluating and comparing the quality of several sound texture synthesis methods. Sound textures are a broad class of sounds ranging from the the noise of the rain falling to the chatter of a crowd, and including many environmental sounds (such as the howling of the wind or the cracking of a fire). Their common point is that they are composed of numerous small audio events that create a sort of sonic background."

This was followed by a presentation of sound texture synthesis methods and their aim:

> "The methods presented in this test are re-synthesis methods: starting from an existing sound texture recording, they attempt at creating a sound that mimics the original without being its copy, as if it had been recorded moments later."

After which the task task they were given was then detailed:

> "For each of the 10 textures presented bellow, you are asked to rate each synthesis method on how close it manages to mimic the style of the original on a scale ranging from 0 (unrecognizable) to 100 (perfect resemblance)."

This was followed by a few recommendations regarding the filling of the test. Participants were also asked to imperatively use earphones or headphones to listen to the sound samples (given participants' tendency to not read instructions, this was also checked at a further point in the test).

Each original was then presented for listening and followed by its re-synthesis by the different methods (including the hidden reference and the anchor). Each sound could be listened to any amount of time. On the same row as each synthesis, a horizontal slider allowed the participants to rate the sound: the numerical value (comprised between 0 and 100) corresponding to the slider position was displayed immediately next to it. This slider was by default at a neutral middle position, corresponding to a value of 50.

In order to avoid introducing any bias, the names of the textures as well as the names of the methods at the origin of each synthesized sound were omitted: instead, they were replaced by un-telling names like "texture 1" or "method 3", in which the number was simply an indication of position within the test. In the case of texture names, this was done to avoid the risk of participants rating the closeness of each synthesis to the idea her/he could have of "the rain", or "applauses" instead of the original. In the case of method names, this was done to avoid having participants remembering the results of a method on previous textures and forming pre-conceived judgements.

Both the order of the methods and that of the textures were also randomized, respectively within a same test and between several tests. Much like for the anonymization of their names, the randomization of the order of the methods was done to avoid having participants recognize methods by their position. The randomization of the order of textures was done to avoid bias on a broader scale. It was likely that the behavior of participants toward the textures they were listening to would evolve throughout the evaluation: on the one hand they might tire and fill the test more hastily toward its end, while on the other they might undergo a learning process. This process could for instance be the learning of some discriminating characteristics that allowed them to judge more easily the difference between a synthesis texture and its original. Both effects would cause the participants to rate the last textures differently than the firsts, hence our decision to shuffle the order of the textures between each test and thus spread this effect more evenly among textures.

A showcase of the web-page of the evaluation is given in Figure 7.1.

After the test, participants were also asked a number of questions relating to themselves: those included their age, the means they had used to listen to the sound samples, their familiarity with sound textures and listening tests as well as if they worked in audio-related fields. Those were meant as criteria by which the results of the tests could be filtered. After being prompted to check that all textures had been evaluated, the participants then had the option to finish their test and send their results.

7.2.2 Results analysis

A total of 69 evaluations were filled over the course of a few weeks. Out of those, 1 was excluded for leaving most ratings untouched, and 4 for not using headphones or earphones: this led to a total of 64 valid evaluations.

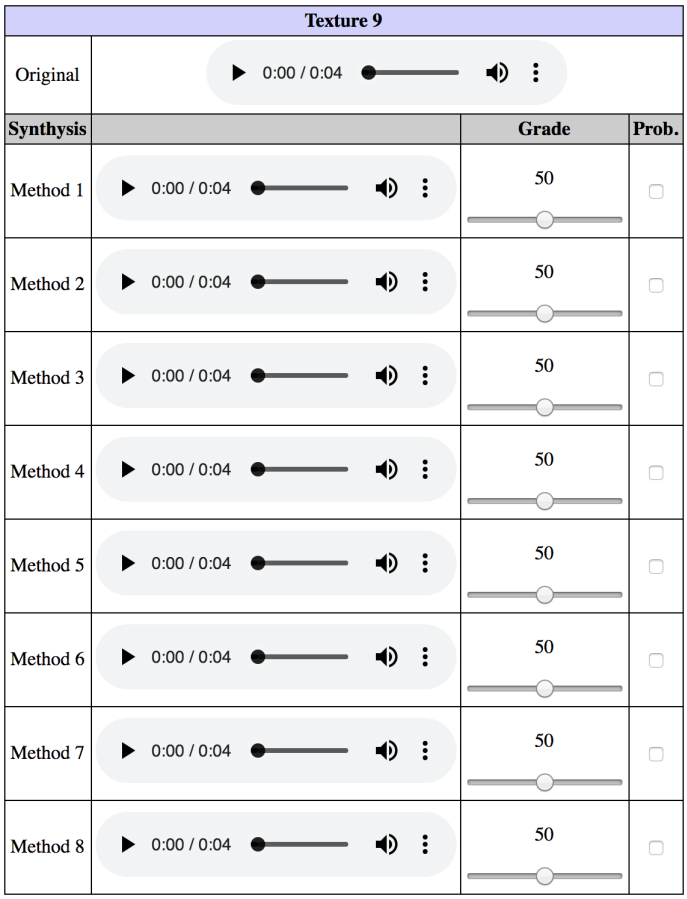


Figure 7.1: *Example of a default section of the evaluation: the original sound is followed by the sounds synthesized using the methods to be compared. The grade attributed to each sound may range from 0 to 100, and is set using a slider. Additionally, any problem with a sound may be signaled by ticking the box "Prob.".*

From one participant to another, different rating behaviors can be observed: some people use only the topmost range of available grades with very little differences between the different methods, while others systematically put the lowest rated method at 0 with grades spanning the whole available range. We find that the rankings of the different method for each texture are more telling and more stable than grades across participants, and as such use those as main metric. The rankings range from 1 (preferred) to 8 (rejected), and an average ranking is given when several method have the same grade.

We use box plots as a way to display data without making any assumption regarding its statistical distribution. The lower, middle and upper section of a box respectively represent the 1st quartile, median and 3rd quartile[25] of the numerical data being represented. As such, the middle section of a box being closer to one edge indicates that the data is skewed in this direction. The whiskers extending from the box show the extent of the rest of the data, apart from outliers which are represented as dots. Those outliers are defined as numerical data that is not within

[25]The 1st quartile splits the lowest 25% of a numerical data from the highest 75%, the median the lowest 50% from the highest 50% and the the 3rd quartile the lowest 75% from the highest 25%.

1.5 interquartile range[26] of the 1st or 3rd quartile. We color the hidden reference and anchors in white, our methods in shades of red and other state of the art methods in shades of blue.

For compactness' sake and within this chapter, the name of the methods are abbreviated to: *mcdermott* for the method introduced in [McDermott and Simoncelli 2011], *ulyanov* for the one introduced in [Ulyanov et al. n.d.], *antognini* for the one introduced in [Antognini et al. 2018], *mcderm_imp* for our extension of McDermott & Simoncelli's method presented in Chapter 4, *spec* for our CNN-based method working on log-spectrograms introduced in Chapter 5 and *RI* for our CNN-based method working on RI spectrograms introduced in Chapter 6.

Global analysis

Global rankings of the different methods across all textures are displayed on Figure 7.3.

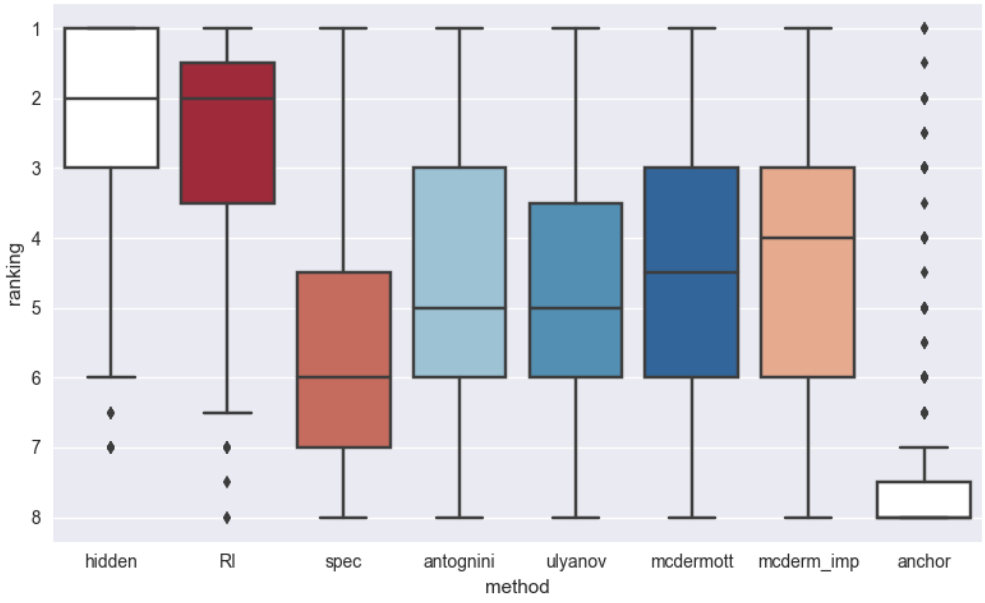


Figure 7.2: *Rankings of the different methods across all textures: hidden reference and anchors are colored in white, our methods in shades of red and state of the art methods in shades of blue.*

The high rankings of the hidden reference, shown by its high median, are encouraging as they show that participants correctly understood the task given to them. The rankings of *RI* being close to those of the hidden reference is an extremely positive result. The difference between these rankings and those of *spec* is also a concrete proof of the improvements that the changes in CNN architecture and time-frequency representation bring. As per our own observation, there seem to be little difference between the rankings of *mcdermott* and our variant using time domain imposition, *mcderm_imp*.

[26]Interquartile range (IQR) is the distance between the 1st and 3rd quartiles.

However, those results represent averages across all textures and may hide more intricate behaviors. The following is thus a selection of the more interesting results across those different textures.

Analysis of "fire"

Rankings of the different methods for the texture "fire" are shown in Figure **??**, while the corresponding audio signals are available at `http://recherche.ircam.fr/anasyn/caracalla/thesis/eval.php`.

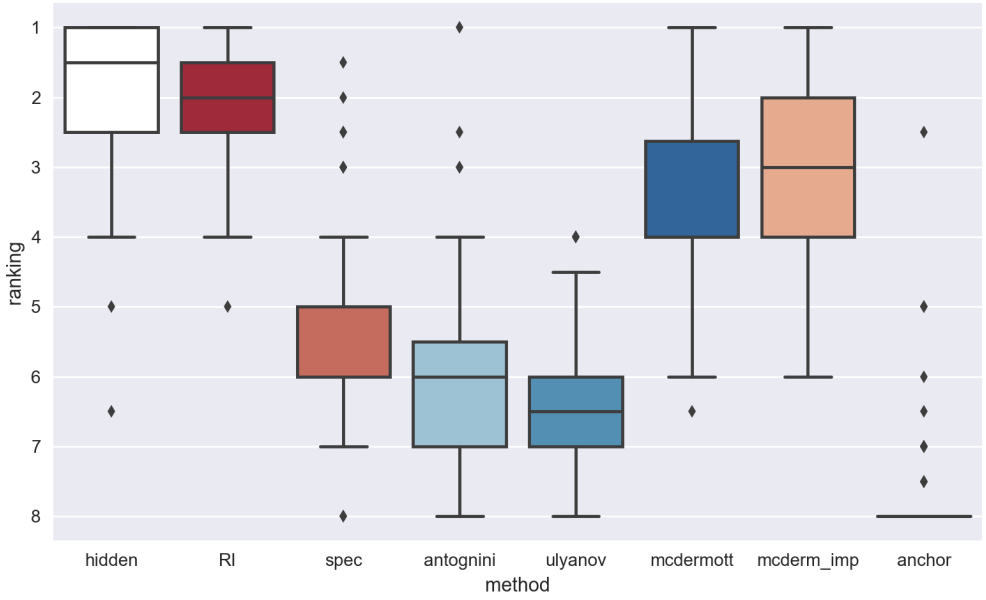


Figure 7.3: *Rankings of the different methods for the texture "fire": hidden reference and anchors are colored in white, our methods in shades of red and state of the art methods in shades of blue.*

This texture contains both a low frequency rumbling and sharp impacts covering most of the spectrum and caused by the fire cracking. It is expected that the filtered white noise of the anchor would be easily discriminated and ranked last. While *mcdermott* and *mcderm_imp* manage to synthesize the rumbling well, their impacts (as observed in Section 4.2.3) are only mildly convincing. These impacts are even more poorly recreated by CNN-based methods working on spectrograms (*antognini*, *ulyanov* and *spec*), due to them not being able to reproduce the high correlations across phases: they place right above the anchor and behind *mcdermott* and *mcderm_imp*. *RI* ranks similarly to the hidden reference: by implicitly containing the phase information, the RI representation appears to be suited to impact synthesis.

Results for the texture "sink", which also contains a great number of sharp events spanning most of the spectrum, are similar to those of the texture "fire".

Analysis of "insects"

Rankings of the different methods for the texture "insects" are shown in Figure 7.4, while the corresponding audio signals are available at `http://recherche.ircam.`

fr/anasyn/caracalla/thesis/eval.php.

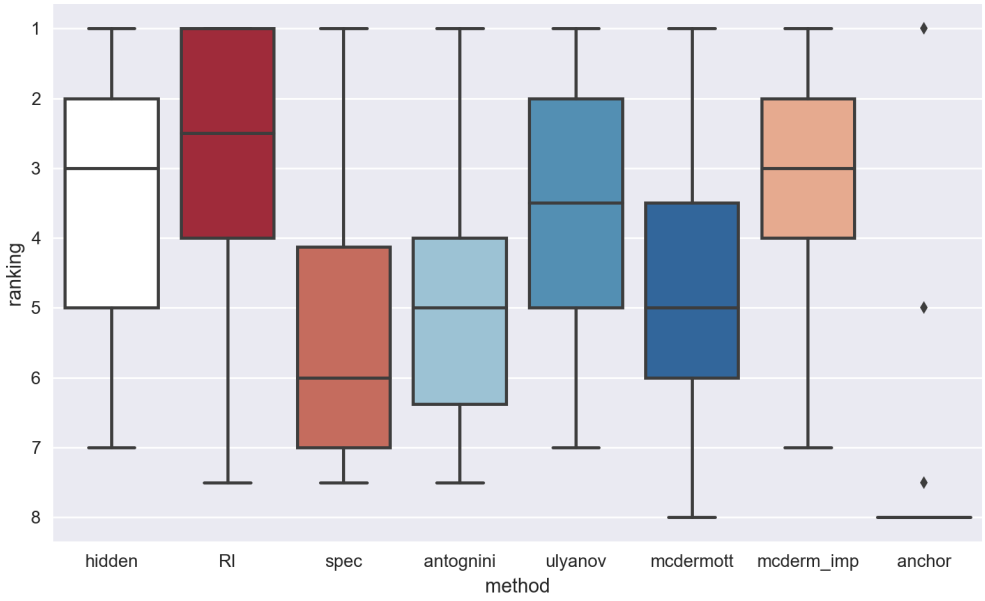


Figure 7.4: *Rankings of the different methods for the texture "insects": hidden reference and anchors are colored in white, our methods in shades of red and state of the art methods in shades of blue.*

This texture is more complex and contains several overlapping elements. Some insects are noisily chirping at a fast rate in the upper frequency bands, while others are chirping more rhythmically and slowly at a medium pitch. In addition to them, a constant noise is audible throughout all of the texture at a lower pitch. Taking the complexity of the texture into account, it is understandable that the rankings of the different methods are less determinate than for the simpler texture "fire": compared to its results, the rankings for each method span broader ranges and the medians are closer. Although the anchor appears to be easy to discriminate, the hidden reference is notably not the highest ranked method.

Because the noisy high frequency chirps are well-reproduced by all methods, the best ranked ones (being *RI*, *ulyanov* and *mcderm_imp*) are those that manage to convincingly reproduce the rhythmical, mid-frequency chirps. *spec* contains an audible and distinctive noise which is probably a consequence of the poor reproduction of said chirp: this seems to be responsible for its poor results in this case.

Analysis of "crowd"

Rankings of the different methods for the texture "crowd" are shown in Figure 7.5, while the corresponding audio signals are available at `http://recherche.ircam.fr/anasyn/caracalla/thesis/eval.php`.

This texture contains both a mid-frequency hubbub and high-pitched salient events, probably due to cutlery being used. *spec*, *ulyanov*, *mcdermott* and *mcderm_imp* are similarly ranked last of the synthesis methods (outside of the anchor). In the case of *spec* and *ulyanov*, this is due to an important noisiness of the synthesized sounds. In

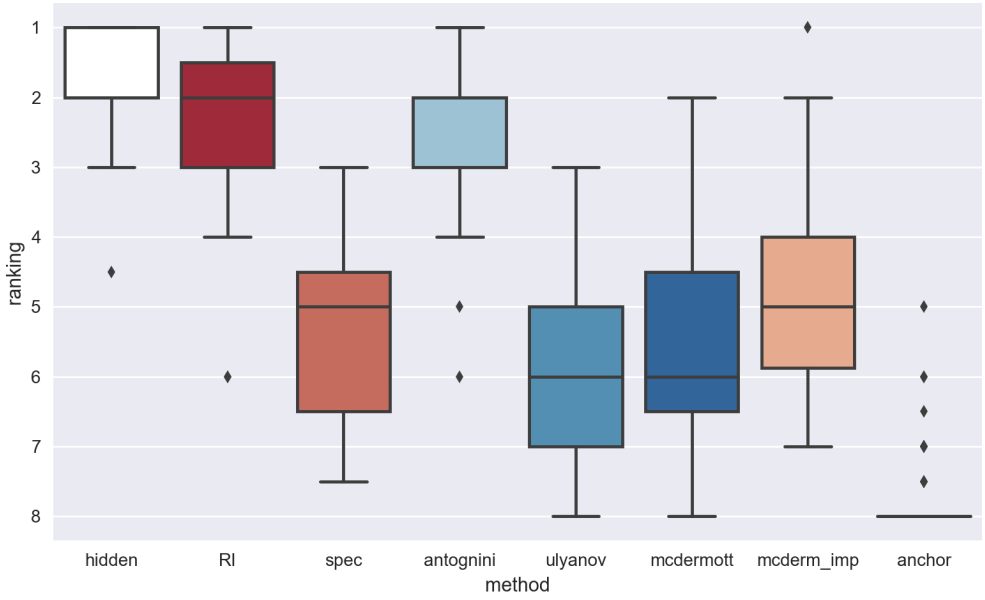


Figure 7.5: *Rankings of the different methods for the texture "crowd": hidden reference and anchors are colored in white, our methods in shades of red and state of the art methods in shades of blue.*

the case of *mcdermott* and *mcderm_imp*, this is due to the presence of salient events: being too spread apart they are not well represented by a statistical description, which impacts the results of the synthesis. Because these salient events are not too sharp, the phases of the different frequency bins do not need to be as correlated as for impacts: they are recreated convincingly by *antognini* despite the representation used not taking phase into account. *RI* ranks similarly well, just behind the hidden reference.

Analysis of "wind"

Rankings of the different methods for the texture "wind" are shown in Figure 7.6, while the corresponding audio signals are available at `http://recherche.ircam.fr/anasyn/caracalla/thesis/eval.php`..

This textures contains the howling of the wind, which can be described as a pitched noise which pitch evolves through time. Oddly enough, the anchor is clearly not the worst ranked method in this case: because the howling resembles a filtered white noise, the anchor is convincing enough to be rated higher than most methods (including all of our own) despite having a constant pitch. However, we do not have a definitive explanation for the important difference in behavior between *mcdermott* and *mcderm_imp*: this may either be due to the original statistics imposition (as presented in [McDermott and Simoncelli 2011]) fitting noisy textures better, or to a difference in the optimization method parameters.

This texture is the only one for which *RI* is not ranked high, although this phenomenon can be explained. Due to the size of the filters of the CNN described in Section 6.4.2, the characteristic size of the events it may reproduce is approximately of 0.5 seconds: the texture "wind", however, is the only one that contains an event

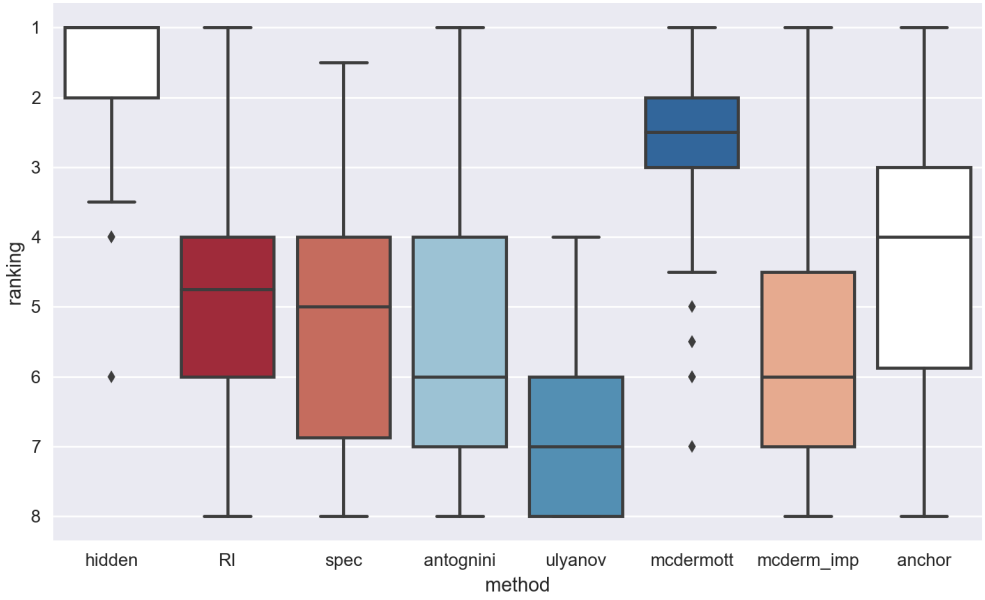


Figure 7.6: *Rankings of the different methods for the texture "wind": hidden reference and anchors are colored in white, our methods in shades of red and state of the art methods in shades of blue.*

(in this case, the howling), which characteristic time is of several seconds. As a result, the texture synthesized by *RI* does not manage to reproduce the slow evolution of the howling. This behavior could be changed by horizontally extending the filters of its CNN, although doing this would mean reproducing longer patches of the original: we would thus risk creating a synthesized sound resembling the original texture too much.

The results of *RI*, our updated synthesis method presented in Chapter 6, are overall very positive: outside of textures evolving at a slow pace, it manages to reproduce all textures similarly well and with almost the same rankings as the hidden reference. Given its good results on a wide array of textures, it also confirms the flexibility of our method: this only leaves its variability to be asserted.

7.3 Variability evaluation

Having the outputs of a texture synthesis algorithm be too similar between themselves (or with the original texture in the case of re-synthesis methods) entirely defeats the purpose of texture synthesis: our aim when evaluating variability is thus to make sure that the sounds synthesized by our *RI* method are not simple duplicates of their original textures, and thus that two sounds synthesized using the same original are also different from one another. Because this matter involves a quantity of subjective decisions regarding what passes as "too similar", our aim is mainly to provide data that quantifies this similarity and to discuss it.

7.3.1 Evaluation preparation

The simplest idea when comparing how similar two sounds are would be to compute their cross-correlations. However, this method of displaying similarity is not adapted

to textures: we expect most synthesized sounds to only be similar to the original texture on a local scale, and cross-correlations might drown this phenomenon within the sum of the dot product between the two (shifted) signals. It is thus more fitting to locally compare both signals via the computation of a similarity matrix, although this still requires a choice of perceptual distance.

Because of the sensibility of temporal signals to phase variations that leave the perceived sound unchanged, using the euclidean distance between them as a perceptual distance is ill-suited. Instead we use the cosine similarity[27] between spectrogram frames: although we are conscious that a number of perceptual properties are not taken into account with such a simple distance, it is however complete enough for the sake of variability evaluation. The choice of using the cosine similarity instead of the euclidean distance is motivated by its independence from the norm of the compared elements, making the similarity matrix more even and easy to read.

By computing the similarity matrix between the spectrograms of both original and synthesized sounds we thus create an understandable and objective representation of the local similarities between textures.

7.3.2 Results analysis

We compute said matrices for two original sounds previously used in the realism evaluation, "fire" and "crowd". For comparison's sake, the similarity of the originals with themselves and with the *antognini* and *mcderm_imp* methods are computed as well. The resulting matrices are displayed on Figure 7.7.

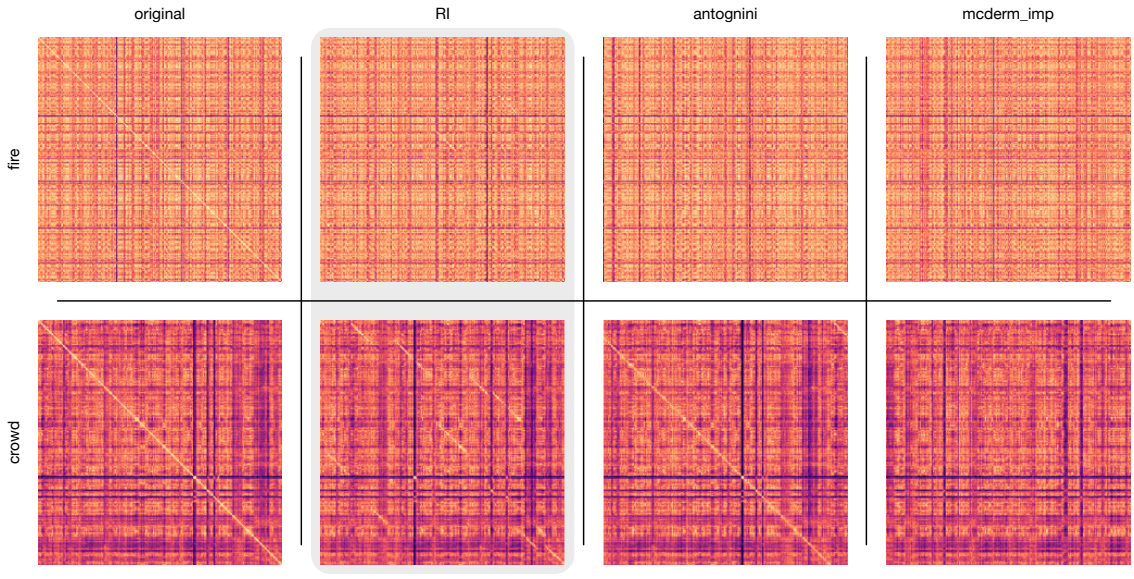


Figure 7.7: *Similarity matrices between the spectrograms of two original texture (left) and the corresponding texture synthesized using different methods (top).*

[27] As a reminder, the cosine similarity between two vectors a and b is defined as $\frac{a.b}{\|a\|\|b\|}$, with . the element-wise product and $\|\|$ the euclidean norm.

The origin of those matrices are at the top left: the vertical axis is the index of the original spectrogram frame, while the horizontal axis is the index of the synthesized spectrogram frame. High values are associated with lighter colors, and all matrices use a common color-map in order for our visual comparison to be coherent.

The apparent diagonal on the similarity matrix of the originals with themselves are expected: each frame is perfectly identical to itself. However, the fact that the texture "fire" is of an overall lighter tone than the texture "crowd" is telling: this means that the frames of its spectrogram are also similar between themselves, which is to be expected since "fire" is more monotonous than "crowd". The "crowd" similarity matrix of *antognini* presents a lightly colored shifted diagonal, meaning that this texture is similar to the original one (albeit slightly shifted in time). This might come from the fact that the filters used in the CNN of *antognini* span large portions of the time-frequency domain, and may constrain large portions of the synthesized spectrogram to be identical to that of the original. On the contrary, both "fire" and "crowd" of *mcderm_imp* are devoid of any diagonal: this means that this method does not reproduce any portion of the original. This is due to the fact that it uses a parametrization with much fewer parameters than the CNN-based methods, and is thus less constrained.

Our method *RI* appears to behave intermediately. In "crowd", diagonal fragments are apparent and indicate that each portion of the synthesized texture is relatively similar to one in the original texture. In "fire", those diagonals are harder to perceive, albeit at least partially present. To verify this, we display the position of the highest value for each column of both similarity matrices and its value in Figure 7.8. While the position of the maximum gives us an estimate of which frame in the original resemble a given frame in our synthesized texture, its value illustrates the degree of resemblance between the two.

This representation highlights the presence of diagonals in "crowd" (as is also noticeable on the simple similarity matrix), but also in "fire"[28]. This means that even in our synthesized "fire" texture, segments of the original are reproduced: given that the maximum similarity score is more erratic than in "crowd", this reproduction seems however slightly less exact. Since diagonals appear not to overlaps with regard to the vertical axis, it would seem that each segment is also reproduced exactly once. In light of this, our approach could be compared to a granular synthesis method: because the diagonals in our similarity matrix have varying sizes, *RI* may be said to indirectly selects segments of the original texture of various sizes and smoothly reorders them to recreate its synthesis texture.

This interpretation is however not completely exact. First, removing the tall filters of the CNN used in *RI* strongly attenuates the diagonal values and reduces the exactitude of the reproduction, albeit to the known cost of degrading the quality of impact re-synthesis. Additionally, and since *mcderm_imp* presents none of this behavior, a lighter parametrization (in the sense of using less parameters) should also attenuate the exactitude of the reproduction. However, such lighter parametrizations

[28] A sharp eye might notice that small top-left and bottom-right diagonals are present in both: those correspond to the unexplained border effect mentioned in Section 6.1.2.

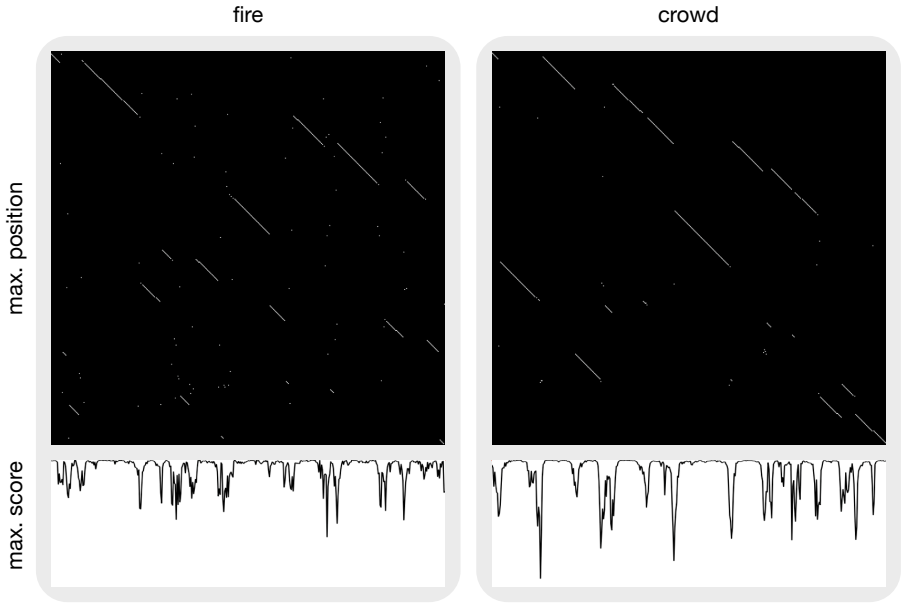


Figure 7.8: *Top: position of the highest value for each column in the similarity matrices between a texture "fire", a texture "crowd" and textures synthesized using our method. Bottom: value of this maximum displayed between 0.75 and 1.*

are known not to be suited to the reproduction of salient events: the local similarities between synthesized textures and original ones is thus a consequence of our algorithm being designed to re-synthesize these occasional events.

References for chapter 7

Antognini, J., Hoffman, M., and Weiss, R. J. (2018). "Synthesizing diverse, high-quality audio textures". In: *arXiv preprint arXiv:1806.08002* (cit. on pp. 43, 84, 85, 88, 92, 96, 100, 106–108, 112, 123).

ITU-R, I. T. U. R. C. A. (2003). "Method for the subjective assessment of intermediate quality levels of coding systems (ITU-R BS.1534-1)." In: (cit. on p. 107).

McDermott, J. H. and Simoncelli, E. P. (2011). "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis". In: *Neuron* (cit. on pp. 42, 48, 49, 52, 54, 55, 60, 62, 63, 65, 71, 82, 85, 88, 107, 108, 112, 115, 122, 123).

Ulyanov, D. and Lebedev, V. (n.d.). *Audio texture synthesis and style transfer* (cit. on pp. 43, 83–85, 88, 91, 92, 107, 108, 112, 123).

Chapter 8

Conclusion

Chapter overview

Concluding this thesis, a list of our contributions to sound texture synthesis is made. Possible future works are also evoked in order to outline potential directions toward which parametric sound texture synthesis might be taken.

Contents

The work presented in this thesis focuses on parametric synthesis methods for sound textures. More precisely, the parameters used in the investigated synthesis algorithms are summary statistics: be they hand-crafted to fit a perceptual model or left for a CNN to extract, they describe textures by a series of statistics that are then imposed onto a base sound. This imposition is not as direct as the term may appear, since the base signal is iteratively modified until its own statistics match that of a target texture. Under the assumption that the parametrization is fitting for textures, the generated sound is perceptually similar to the target texture while still behaving somehow unpredictably.

Because sound textures are hard to categorize, methods that aim at synthesizing them are left in an in-between area: should they suppose that the target texture is perfectly random, or should they presume that occasional salient events might be present ? Our goal at the start of our work was to create an algorithm that would work with a broad definition of textures, and we hope that our results go along this direction. However, and despite the extremely positive results of our evaluation on the realism of our synthesized textures, we believe that much work is left in this domain.

The present chapter is a conclusion to this thesis, and as such aims at both summarizing our work and pointing toward promising leads in sound texture synthesis.

8.1 Contributions overview

Our contributions to parametric sound synthesis are rather varied, and range from the use of a mathematical paradigm like Wirtinger calculus to the more experimental investigations of the process at work in CNN-based texture synthesis: the major ones are listed in the following sections.

8.1.1 Time domain imposition of time-frequency statistics

Given a cost or loss function in an optimization problem (such as the training of a neural network, or in our case the imposition of statistics onto a signal), the gradient of the function with respect to the signal is in practice often computed using a chain rule. This chain rule gives the derivative of a composition of functions as a combination of their own gradients or Jacobian matrices. This supposes that involved functions are differentiable on their domain: this is not the case of the DFT.

Using an extension of $\mathbb{C}$-differentiability, we have demonstrated that a there exists an efficient way of expressing the gradient of a cost function even if it involves the composition of a DFT or inverse DFT. This work resulted in a conference paper: [Caracalla et al. 2017]. We applied this method in order to compute the gradient of a distance computed from time-frequency statistics. This allowed us to circumvent the needlessly complex imposition method proposed in [McDermott and Simoncelli 2011] and perform an imposition of all statistics at once and directly onto the time signal. In addition to being useful in this particular case, we have also kept this paradigm of imposing statistics directly in the time domain when investigating CNN-based synthesis methods.

8.1.2 Investigation of CNN-based visual texture synthesis

Parametric texture synthesis using CNN-based statistics, as presented in [L. Gatys et al. 2015], produces impressive visual results. We feel however that, given that it lacks the perceptual motivation of methods such as the one introduced in [Portilla et al. 2000], the interpretation of its inner working is not explored enough.

By iteratively creating images to which filters or correlations of filters react strongly to, or "identikit", we have proposed an intuitive interpretation of the parameters contained in the Gram matrices that stores the synthesis parameters. This interpretation is that of an "amount of pattern presence": this pattern is implicitly described by a filter and is of a size similar to its ERF. The image to which a correlation of filter reacts strongly may be seen as a hybrid of the identikit of each filter. Through experimenting on CNN-based synthesis, we have also shown that this interpretation is coherent with the practical behavior of this synthesis and allows us to anticipate the consequences of design choices on the synthesis algorithm.

8.1.3 Design of a CNN-based sound texture synthesis algorithm

Due to the fundamental differences between sound and image, the CNN-based algorithm presented in [L. Gatys et al. 2015] cannot be directly applied to the time-frequency representation of a sound texture. Instead, we have proposed a new sound texture algorithm inspired by it. Starting from a time-frequency representation, we use a series of untrained CNN with a single convolutional layer each to extract a parametrization from it. The filters of those CNNs have sizes chosen to best represent the events commonly contained in time-frequency representations of textures, and the parametrization is chosen so as to only be time-invariant.

Contrarily to existing methods that use spectrograms as image, we have shown that using the compressed real and imaginary (RI) parts of the STFT as the two colors channels of an artificial image leads to more convincing results, rid of artefacts commonly encountered in methods that discard the phase of the STFT. In line with our previous work, we also impose the parameters of the target texture onto the base signal directly in the time domain: this prevents us from creating an incoherent STFT as the output of our synthesis algorithm. The beginning of this work led to a conference paper: [Caracalla et al. 2019].

8.1.4 Evaluation of both realism and variability

Evaluating the overall quality of a sound texture synthesis is a complex task, mainly due to how subjective such ratings may be. In order to obtain more objective results we have created an online perceptual test based on MUSHRA. In this test, participants were asked to evaluate how close a selection of methods sounded to an array of original textures. Those methods included the parametric algorithms presented in [McDermott and Simoncelli 2011], [Ulyanov et al. n.d.] and [Antognini et al. 2018], 3 of our own, but also a hidden references (the continuation of the target texture) and an anchor (a filtered white noise) for comparison's sake. The results of those methods reveal different behaviors between each methods, but are overall clearly positive for our latest CNN-based synthesis using RI representation.

Additionally, we have also created a more quantitative evaluation of the similitude between a target texture and a synthesized texture. This evaluation was done by computing the similarity matrix between the frames of spectrograms of both sounds. This similarity matrix may be processed even further to reveal the reproduction of segments of the original in the synthesized sound, and may be used as a way to make sure that the algorithm is behaving as intended.

8.2 Future works

Our work has naturally revealed a few areas in sound texture synthesis that we deem interesting enough to investigate: the following sections are a selection of the more pertinent of those.

8.2.1 Investigation of the border effects

Whatever the representation used, our CNN-based synthesis algorithm presents border artefacts: the first few and last frames of the representation of the synthesized sound are extremely similar to that of the original texture. This phenomenon is particularly visible on the similarity matrix between the two. Although the actual consequences of this effect are easy to avoid by slightly cropping the synthesized sound, the fact that we are not able to explain it is a evidence of the fact that we do not completely understand the working of our algorithm yet. As such, its investigation would surely bring more insights on our synthesis algorithm.

8.2.2 Improved parametrization

The parametrization we are using as of now is assuredly too heavy-handed: our current version uses parameter matrices storing almost 30 millions parameters (or statistics), despite working on sounds that are typically less than 1 million samples long. It is thus certain that many of those parameters are redundant and could be left out of the parametrization. Reducing the number of parameters would also considerably fasten computation times: as of now, the process is around 60 times slower than real time.

To do so, a few methods are conceivable. One could first directly chose to only use a selection of parameters: this selection could be performed by discarding parameters that are systematically constant on an array of textures, or that are extremely correlated to other parameters. Another way of proceeding would be to act on a lower (algorithm-wise) level, notably the filters of the CNN. Since we have a better understanding of their role, it should now be possible to only use a set of pertinent filters. The random draw of their weights has so far been a way of easily initializing the filters: however, initializing them so that they efficiently cover a wide array of shapes seems more appropriate, and a possibility worth investigating. Additionally, reducing the number of filters would quadratically affect the number of parameters and thus be the most efficient way of reducing the size of the parametrization.

8.2.3 Attempts at audio style transfer

In [L. A. Gatys et al. 2016], the authors succeed in transferring the style of style image onto an input image: this results in an output image that has the same high-level content as the input (in the sense that one may still recognize the elements present

in it) while having the local textural properties of the style image. This is done by imposing the parameters of the visual texture parametrization introduced in [L. Gatys et al. 2015] onto the input image, albeit only for the first layer of the deep CNN.

Such a concept might be adaptable to the audio domain. One could for instance impose the parameters of given sound texture onto a base sound, but while only using parameters extracted by small filters: parameters extracted by bigger filters could be taken from another sound texture, in order to study the effect of imposing the broad organization of a texture and the local patterns of another. This idea is merely an example of available possibility: artistic usages and potential re-adaptations of our algorithm are plentiful.

8.2.4 Investigation of synthesis control

Our last proposition relates to an area of sound texture synthesis that we regret no having had time to explore further. The ability to extend indefinitely a synthesized texture and to control its properties are crucial in current uses of sound texture synthesis. Although our algorithm is now able to convincingly create new textures from an original one, its lack in regard to those two abilities severely affects its potential uses.

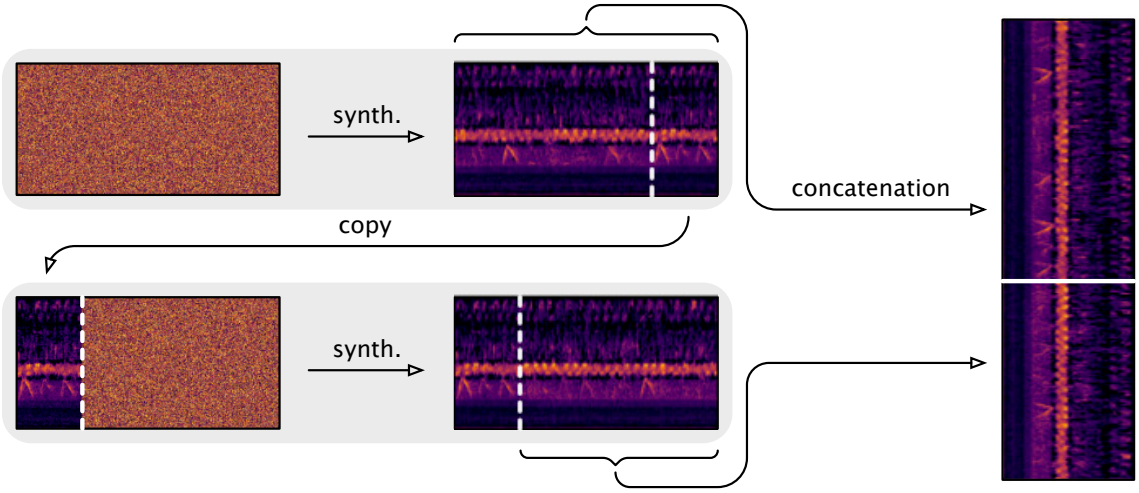


Figure 8.1: *Extension of a first synthesis by copying its end onto the start of a white noise signal. This signal is then used as initialization of another synthesis, while preventing the common part from being modified. The tiles are then concatenated to form a longer texture. Sound signals are represented by their logspectrogram for explanation's sake.*

However, we strongly believe that the paradigm of parametric synthesis is powerful enough to allow for both extensibility and control. In the case of the first, we have already started investigating a process inspired by the "exquisite cadaver" game, where, one has to continue the drawing of someone else while only being able to see the borders of the other's drawing. In our case we first synthesize an initial sound texture from a given target, and copy the end of the synthesized sound onto the start of a white noise signal. We then perform another synthesis using this noise-texture hybrid signal as initialization (keeping the same target), while preventing the op-

timization from being performed on the section the copied section. This results in a continuous texture seamlessly extending on the copied part, thus being able to perfectly follow where the previous synthesis left off. We only need to concatenate the newly generated texture to the previous one to create a longer sound texture. This process can obviously be repeated any number of times so as to obtain a texture of any desired length. It is illustrated in Figure 8.1.

By changing target texture at each step of this "exquisite synthesis", or using interpolated parameter matrices, one could imagine being able to generate a texture that evolves from one sound to another, thus granting a certain degree of control other the algorithm. This is merely a thought experiment for now, but it illustrates the potential strength of parametric texture synthesis: by projecting sounds onto a parameter space in which similar textures are close together, it might be possible to explore said space to grant control over the synthesized texture.

References for chapter 8

Antognini, J., Hoffman, M., and Weiss, R. J. (2018). "Synthesizing diverse, high-quality audio textures". In: *arXiv preprint arXiv:1806.08002* (cit. on pp. 43, 84, 85, 88, 92, 96, 100, 106–108, 112, 123).

Caracalla, H. and Roebel, A. (2017). "Gradient conversion between time and frequency domains using wirtinger calculus". In: *Digital Audio Effects (DAFx)* (cit. on pp. 62, 82, 122).

– (2019). "Sound texture synthesis using convolutional neural networks". In: *Digital Audio Effects (DAFx)* (cit. on pp. 79, 123).

Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). "Image style transfer using convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (cit. on pp. 42, 70, 124).

Gatys, L., Ecker, A. S., and Bethge, M. (2015). "Texture synthesis using convolutional neural networks". In: *Advances in neural information processing systems* (cit. on pp. 42, 48, 70, 72–74, 76, 79, 82, 83, 90, 92, 101, 123, 125).

McDermott, J. H. and Simoncelli, E. P. (2011). "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis". In: *Neuron* (cit. on pp. 42, 48, 49, 52, 54, 55, 60, 62, 63, 65, 71, 82, 85, 88, 107, 108, 112, 115, 122, 123).

Portilla, J. and Simoncelli, E. P. (2000). "A parametric texture model based on joint statistics of complex wavelet coefficients". In: *International journal of computer vision* (cit. on pp. 42, 50, 70, 71, 73, 123).

Ulyanov, D. and Lebedev, V. (n.d.). *Audio texture synthesis and style transfer* (cit. on pp. 43, 83–85, 88, 91, 92, 107, 108, 112, 123).

Chapter 9

Bibliography

Antognini, J., Hoffman, M., and Weiss, R. J. (2018). "Synthesizing diverse, high-quality audio textures". In: *arXiv preprint arXiv:1806.08002* (cit. on pp. 43, 84, 85, 88, 92, 96, 100, 106–108, 112, 123).

Athineos, M. and Ellis, D. P. (2003). "Sound texture modelling with linear prediction in both time and frequency domains". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cit. on pp. 2, 40).

Attias, H. and Schreiner, C. E. (1997). "Temporal low-order statistics of natural sounds". In: *Advances in neural information processing systems* (cit. on p. 41).

Bar-Joseph, Z. et al. (2001). "Texture mixing and texture movie synthesis using statistical learning". In: *IEEE Transactions on visualization and computer graphics* (cit. on p. 39).

Barry, S. and Kim, Y. (2018). *"Style" Transfer for Musical Audio Using Multiple Time-Frequency Representations* (cit. on pp. 43, 82).

Blaauw, M. and Bonada, J. (2016). "Modeling and transforming speech using variational autoencoders." In: *Interspeech* (cit. on p. 23).

Bouboulis, P. (2010). "Wirtinger's calculus in general Hilbert spaces". In: *arXiv preprint arXiv:1005.5170* (cit. on pp. 56, 57).

Brandwood, D. (1983). "A complex gradient operator and its application in adaptive array theory". In: *IEE Proceedings F-Communications, Radar and Signal Processing* (cit. on p. 57).

Caracalla, H. and Roebel, A. (2017). "Gradient conversion between time and frequency domains using wirtinger calculus". In: *Digital Audio Effects (DAFx)* (cit. on pp. 62, 82, 122).

– (2019). "Sound texture synthesis using convolutional neural networks". In: *Digital Audio Effects (DAFx)* (cit. on pp. 79, 123).

Chadwick, J. N. and James, D. L. (2011). "Animating fire with sound". In: *ACM Transactions on Graphics (TOG)* (cit. on p. 37).

Choi, K. et al. (2017). "Convolutional recurrent neural networks for music classification". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cit. on p. 23).

Cohen-Hadria, A. et al. (2019). "Voice Anonymization in Urban Sound Recordings". In: *29th IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. Pittsburgh, Pennsylvania, USA (cit. on p. 12).

Di Scipio, A. (1999). "Synthesis of environmental sound textures by iterated nonlinear functions". In: *Proceedings of the 2nd COST G-6 Workshop on Digital Audio Effects (DAFx)* (cit. on p. 40).

Doras, G., Esling, P., and Peeters, G. (2019). "On the use of u-net for dominant melody estimation in polyphonic music". In: *International Workshop on Multilayer Music Representation and Processing (MMRP)* (cit. on p. 12).

Dubnov, S., Assayag, G., and Cont, A. (2007). "Audio oracle: A new algorithm for fast learning of audio structures". In: (cit. on p. 39).

Dubnov, S., Bar-Joseph, Z., et al. (2002). "Synthesizing sound textures through wavelet tree learning". In: *IEEE Computer Graphics and Applications* (cit. on p. 39).

Efros, A. A. and Freeman, W. T. (2001). "Image quilting for texture synthesis and transfer". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (cit. on p. 39).

Engel, J. et al. (2019). "Gansynth: Adversarial neural audio synthesis". In: *arXiv preprint arXiv:1902.08710* (cit. on p. 98).

Erhan, D. et al. (2009). "Visualizing higher-layer features of a deep network". In: *University of Montreal* (cit. on p. 74).

Fischer, R. F. (2005). *Precoding and signal shaping for digital transmission*. John Wiley & Sons (cit. on p. 57).

Fröjd, M. and Horner, A. (2007). "Fast sound Texture synthesis using Overlap-Add." In: *ICMC* (cit. on pp. 38, 39).

Fu, S.-W. et al. (2017). "Complex spectrogram enhancement by convolutional neural network with multi-metrics learning". In: *IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)* (cit. on p. 99).

Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). "Image style transfer using convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (cit. on pp. 42, 70, 124).

Gatys, L., Ecker, A. S., and Bethge, M. (2015). "Texture synthesis using convolutional neural networks". In: *Advances in neural information processing systems* (cit. on pp. 42, 48, 70, 72–74, 76, 79, 82, 83, 90, 92, 101, 123, 125).

Glorot, X. and Bengio, Y. (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *13th International Conference on Artificial Intelligence and Statistics* (cit. on p. 30).

Griffin, D. and Lim, J. (1984). "Signal estimation from modified short-time Fourier transform". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* (cit. on p. 18).

Grinstein, E. et al. (2018). "Audio style transfer". In: *2018 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2018. Proceedings.(ICASSP'18)*. (Cit. on p. 43).

Haralick, R. M. et al. (1979). "Statistical and structural approaches to texture". In: *Proceedings of the IEEE* (cit. on p. 21).

He, K. et al. (2016). "Deep residual learning for image recognition". In: *IEEE conference on computer vision and pattern recognition* (cit. on p. 23).

Heeger, D. J. and Bergen, J. R. (1995). "Pyramid-based texture analysis/synthesis". In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (cit. on p. 50).

Hoffman, M. D. and Cook, P. R. (2006). "Feature-Based Synthesis: Mapping Acoustic and Perceptual Features onto Synthesis Parameters." In: *ICMC* (cit. on pp. 40, 41).

ITU-R, I. T. U. R. C. A. (2003). "Method for the subjective assessment of intermediate quality levels of coding systems (ITU-R BS.1534-1)." In: (cit. on p. 107).

Julesz, B. (1962). “Visual pattern discrimination”. In: *IRE transactions on Information Theory* (cit. on p. 42).

Karras, T., Laine, S., and Aila, T. (2019). “A style-based generator architecture for generative adversarial networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition* (cit. on p. 23).

Kim, H.-S. and Smith, J. (n.d.). “Synthesis of sound textures with tonal components using summary statistics and and all-pole residual modeling”. In: (cit. on pp. 42, 66).

Le Roux, J. et al. (n.d.). “Fast signal reconstruction from magnitude STFT spectrogram based on spectrogram consistency”. In: (cit. on p. 18).

LeCun, Y. et al. (1998). “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* (cit. on pp. 12, 28).

Liao, W.-H. (2015). “Modelling and transformation of sound textures and environmental sounds”. PhD thesis (cit. on pp. 42, 48).

Lu, L., Wenyin, L., and Zhang, H.-J. (2004). “Audio textures: Theory and applications”. In: *IEEE transactions on speech and audio processing* (cit. on pp. 6, 38).

Luo, W. et al. (2016). “Understanding the effective receptive field in deep convolutional neural networks”. In: *Advances in neural information processing systems* (cit. on pp. 75, 76).

Makhoul, J. and Cosell, L. (1976). “LPCW: An LPC vocoder with linear predictive spectral warping”. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cit. on p. 19).

McDermott, J. H., Schemitsch, M., and Simoncelli, E. P. (2013). “Summary statistics in auditory perception”. In: *Nature neuroscience* (cit. on pp. 42, 48).

McDermott, J. H. and Simoncelli, E. P. (2011). “Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis”. In: *Neuron* (cit. on pp. 42, 48, 49, 52, 54, 55, 60, 62, 63, 65, 71, 82, 85, 88, 107, 108, 112, 115, 122, 123).

McWalter, R. I. and Dau, T. (2013). “Analysis of the auditory system via Sound Texture Synthesis”. In: *AIA-DAGA Conference on Acoustics* (cit. on p. 48).

Miklavcic, S. J., Zita, A., and Arvidsson, P. (2004). *Computational real-time sound synthesis of rain*. Department of Science and Technology (ITN), Campus Norrköping, Linköping ... (cit. on p. 36).

Moss, W. et al. (2010). “Sounding liquids: Automatic sound synthesis from fluid simulation”. In: *ACM Transactions on Graphics (TOG)* (cit. on p. 37).

Oksanen, S., Parker, J., and Välimäki, V. (2013). “Physically informed synthesis of jackhammer tool impact sounds”. In: *Digital Audio Effects (DAFx)* (cit. on p. 37).

O’Leary, S. and Röbel, A. (2016). “A montage approach to sound texture synthesis”. In: *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* (cit. on p. 39).

Parker, J. R. and Behm, B. (2004). “Creating audio textures by example: tiling and stitching”. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing* (cit. on p. 39).

Peltola, L. et al. (2007). “Synthesis of hand clapping sounds”. In: *IEEE Transactions on Audio, Speech, and Language Processing* (cit. on p. 37).

Perraudin, N., Balazs, P., and Søndergaard, P. L. (2013). "A fast Griffin-Lim algorithm". In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (cit. on p. 18).

Pons, J., Lidy, T., and Serra, X. (2016). "Experimenting with musically motivated convolutional neural networks". In: *14th International Workshop on Content-Based Multimedia Indexing (CBMI)* (cit. on p. 20).

Portilla, J. and Simoncelli, E. P. (2000). "A parametric texture model based on joint statistics of complex wavelet coefficients". In: *International journal of computer vision* (cit. on pp. 42, 50, 70, 71, 73, 123).

Saint-Arnaud, N. (1995). "Classification of sound textures". PhD thesis. Massachusetts Institute of Technology (cit. on p. 2).

Saint-Arnaud, N. and Popat, K. (1995). "Analysis and synthesis of sound textures". In: *in Readings in Computational Auditory Scene Analysis* (cit. on pp. 4, 38).

Schwarz, D. and O'Leary, S. (2015). "Smooth granular sound texture synthesis by control of timbral similarity". In: (cit. on p. 39).

Simonyan, K. and Zisserman, A. (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (cit. on p. 70).

Smith, J. O. (2007a). *Introduction to digital filters: with audio applications*. Julius Smith (cit. on p. 64).

– (2007b). *Mathematics of the discrete Fourier transform (DFT): with audio applications*. Julius Smith (cit. on p. 13).

Snelgrove, X. (2017). "High-resolution multi-scale neural texture synthesis." In: *SIGGRAPH Asia Technical Briefs* (cit. on p. 97).

Springenberg, J. T. et al. (2014). "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (cit. on p. 28).

Stevens, S. S., Volkmann, J., and Newman, E. B. (1937). "A scale for the measurement of the psychological magnitude pitch". In: *The Journal of the Acoustical Society of America* (cit. on p. 19).

Tomczak, M., Southall, C., and Hockman, J. (2018). "Audio Style Transfer with Rhythmic Constraints". In: *Digital Audio Effects (DAFx)* (cit. on pp. 43, 82).

Ulyanov, D. and Lebedev, V. (n.d.). *Audio texture synthesis and style transfer* (cit. on pp. 43, 83–85, 88, 91, 92, 107, 108, 112, 123).

Ustyuzhaninov, I. et al. (2016). "Texture synthesis using shallow convolutional networks with random filters". In: *arXiv preprint arXiv:1606.00021* (cit. on pp. 79, 80, 82, 90–92).

Wirtinger, W. (1927). "Zur formalen theorie der funktionen von mehr komplexen veränderlichen". In: *Mathematische Annalen* (cit. on p. 56).

Zhou, D. (2006). "Texture analysis and synthesis using a generic Markov-Gibbs image model". PhD thesis. ResearchSpace@ Auckland (cit. on p. 21).

Zwicker, E. (1961). "Subdivision of the audible frequency range into critical bands (Frequenzgruppen)". In: *The Journal of the Acoustical Society of America* (cit. on p. 19).